Towards a formal theory of computability

Helmut Schwichtenberg
(j.w.w. Simon Huber, Basil Karádais)

Mathematisches Institut, LMU, München

Workshop on Constructive Aspects of Logic and Mathematics
Kanazawa, Japan, 8. - 12. March 2010

Finitary algebras as non-flat Scott information systems

- An algebra ι is given by its constructors.
- Examples:

$$0^{\mathbf{N}}, \mathrm{S}^{\mathbf{N}\to\mathbf{N}} \quad \text{for } \mathbf{N} \text{ (unary natural numbers)},$$

$$1^{\mathbf{P}}, S_0^{\mathbf{P}\to\mathbf{P}}, S_1^{\mathbf{P}\to\mathbf{P}} \quad \text{for } \mathbf{P} \text{ of (binary positive numbers)},$$

$$0^{\mathbf{D}} \text{ (axiom) and } \mathrm{C}^{\mathbf{D}\to\mathbf{D}\to\mathbf{D}} \text{ (rule) for } \mathbf{D} \text{ (derivations)}.$$

- Examples of “information tokens”: $\mathrm{S}^n 0$ ($n \geq 0$), $\mathrm{S}^2 *$ (in $\mathbf{N}$), $\mathrm{C}(\mathrm{C}0*)(\mathrm{C}*0)$ (in $\mathbf{D}$) ($*$: special symbol; no information).
- An information token is total if it contains no $*$.
- In $\mathbf{D}$: total token $\sim$ finite (well-founded) derivation.

Finitary algebras as non-flat Scott information systems

- An algebra ι is given by its constructors.
- Examples:

$$0^{\mathbf{N}}, \mathrm{S}^{\mathbf{N}\to\mathbf{N}} \quad \text{for } \mathbf{N} \text{ (unary natural numbers)},$$
$$1^{\mathbf{P}}, S_0^{\mathbf{P}\to\mathbf{P}}, S_1^{\mathbf{P}\to\mathbf{P}} \quad \text{for } \mathbf{P} \text{ of (binary positive numbers)},$$
$$0^{\mathbf{D}} \text{ (axiom) and } \mathrm{C}^{\mathbf{D}\to\mathbf{D}\to\mathbf{D}} \text{ (rule) for } \mathbf{D} \text{ (derivations)}.$$

- Examples of "information tokens": $\mathrm{S}^n 0$ ($n \geq 0$), $\mathrm{S}^2 *$ (in **N**), $\mathrm{C}(\mathrm{C}0*)(\mathrm{C}*0)$ (in **D**) ($*$: special symbol; no information).
- An information token is total if it contains no $*$.
- In **D**: total token $\sim$ finite (well-founded) derivation.

Finitary algebras as non-flat Scott information systems

- An algebra ι is given by its constructors.
- Examples:

$$0^{\mathbf{N}}, \mathrm{S}^{\mathbf{N}\to\mathbf{N}} \quad \text{for } \mathbf{N} \text{ (unary natural numbers)},$$
$$1^{\mathbf{P}}, S_0^{\mathbf{P}\to\mathbf{P}}, S_1^{\mathbf{P}\to\mathbf{P}} \quad \text{for } \mathbf{P} \text{ of (binary positive numbers)},$$
$$0^{\mathbf{D}} \text{ (axiom) and } \mathrm{C}^{\mathbf{D}\to\mathbf{D}\to\mathbf{D}} \text{ (rule) for } \mathbf{D} \text{ (derivations)}.$$

- Examples of "information tokens": $\mathrm{S}^n 0$ ($n \geq 0$), $\mathrm{S}^2 *$ (in **N**), $\mathrm{C}(\mathrm{C}0*)(\mathrm{C}*0)$ (in **D**) ($*$: special symbol; no information).
- An information token is total if it contains no $*$.
- In **D**: total token $\sim$ finite (well-founded) derivation.

Finitary algebras as non-flat Scott information systems

- An algebra ι is given by its constructors.
- Examples:

$$0^{\mathbf{N}}, \mathrm{S}^{\mathbf{N}\to\mathbf{N}} \quad \text{for } \mathbf{N} \text{ (unary natural numbers)},$$
$$1^{\mathbf{P}}, S_0^{\mathbf{P}\to\mathbf{P}}, S_1^{\mathbf{P}\to\mathbf{P}} \quad \text{for } \mathbf{P} \text{ of (binary positive numbers)},$$
$$0^{\mathbf{D}} \text{ (axiom) and } \mathrm{C}^{\mathbf{D}\to\mathbf{D}\to\mathbf{D}} \text{ (rule) for } \mathbf{D} \text{ (derivations)}.$$

- Examples of "information tokens": $\mathrm{S}^n 0$ ($n \geq 0$), $\mathrm{S}^2 *$ (in $\mathbf{N}$), $\mathrm{C}(\mathrm{C}0*)(\mathrm{C}*0)$ (in $\mathbf{D}$) ($*$: special symbol; no information).
- An information token is total if it contains no $*$.
- In $\mathbf{D}$: total token $\sim$ finite (well-founded) derivation.

Finitary algebras as non-flat Scott information systems

- An algebra ι is given by its constructors.
- Examples:

$$0^{\mathbf{N}}, \mathrm{S}^{\mathbf{N}\to\mathbf{N}} \quad \text{for } \mathbf{N} \text{ (unary natural numbers)},$$
$$1^{\mathbf{P}}, S_0^{\mathbf{P}\to\mathbf{P}}, S_1^{\mathbf{P}\to\mathbf{P}} \quad \text{for } \mathbf{P} \text{ of (binary positive numbers)},$$
$$0^{\mathbf{D}} \text{ (axiom) and } \mathrm{C}^{\mathbf{D}\to\mathbf{D}\to\mathbf{D}} \text{ (rule) for } \mathbf{D} \text{ (derivations)}.$$

- Examples of "information tokens": $\mathrm{S}^n 0$ ($n \geq 0$), $\mathrm{S}^2 *$ (in **N**), $\mathrm{C}(\mathrm{C}0*)(\mathrm{C}*0)$ (in **D**) ($*$: special symbol; no information).
- An information token is total if it contains no $*$.
- In **D**: total token $\sim$ finite (well-founded) derivation.

Finitary algebras: consistency, entailment, ideals

For **D** (derivations):

- $\{C0*, C*0\}$ is "consistent", written $C0* \uparrow C*0$.
- $\{C0*, C*0\} \vdash C00$ ("entails").
- Ideals: consistent and "deductively closed" sets of tokens.

Examples of ideals:

- $\{C0*, C**\}$.
- $\{C00, C0*, C*0, C**\}$.
- The deductive closure of a finite (well-founded) derivation.
- $\{C**, C(C**)*, C*(C**), C(C**)(C**), \dots\}$ ("cototal").
- Locally correct, but possibly non well-founded derivations (Mints 1978).

Finitary algebras: consistency, entailment, ideals

For **D** (derivations):

- $\{C0*, C*0\}$ is "consistent", written $C0* \uparrow C*0$.
- $\{C0*, C*0\} \vdash C00$ ("entails").
- Ideals: consistent and "deductively closed" sets of tokens.

Examples of ideals:

- $\{C0*, C**\}$.
- $\{C00, C0*, C*0, C**\}$.
- The deductive closure of a finite (well-founded) derivation.
- $\{C**, C(C**)*, C*(C**), C(C**)(C**), \dots\}$ ("cotototal").
- Locally correct, but possibly non well-founded derivations (Mints 1978).

Finitary algebras: consistency, entailment, ideals

For **D** (derivations):

- $\{C0*, C*0\}$ is "consistent", written $C0* \uparrow C*0$.
- $\{C0*, C*0\} \vdash C00$ ("entails").
- Ideals: consistent and "deductively closed" sets of tokens.

Examples of ideals:

- $\{C0*, C**\}$.
- $\{C00, C0*, C*0, C**\}$.
- The deductive closure of a finite (well-founded) derivation.
- $\{C**, C(C**)*, C*(C**), C(C**)(C**), \dots\}$ ("cototal").
- Locally correct, but possibly non well-founded derivations (Mints 1978).

Finitary algebras: consistency, entailment, ideals

For **D** (derivations):

- $\{C0*, C*0\}$ is "consistent", written $C0* \uparrow C*0$.
- $\{C0*, C*0\} \vdash C00$ ("entails").
- Ideals: consistent and "deductively closed" sets of tokens.

Examples of ideals:

- $\{C0*, C**\}$.
- $\{C00, C0*, C*0, C**\}$.
- The deductive closure of a finite (well-founded) derivation.
- $\{C**, C(C**)*, C*(C**), C(C**)(C**), \dots\}$ ("cototal").
- Locally correct, but possibly non well-founded derivations (Mints 1978).

Finitary algebras: consistency, entailment, ideals

For **D** (derivations):

- $\{\mathrm{C}0*, \mathrm{C}{*}0\}$ is "consistent", written $\mathrm{C}0* \uparrow \mathrm{C}{*}0$.
- $\{\mathrm{C}0*, \mathrm{C}{*}0\} \vdash \mathrm{C}00$ ("entails").
- Ideals: consistent and "deductively closed" sets of tokens.

Examples of ideals:

- $\{\mathrm{C}0*, \mathrm{C}{**}\}$.
- $\{\mathrm{C}00, \mathrm{C}0*, \mathrm{C}{*}0, \mathrm{C}{**}\}$.
- The deductive closure of a finite (well-founded) derivation.
- $\{\mathrm{C}{**}, \mathrm{C}(\mathrm{C}{**})*, \mathrm{C}{*}(\mathrm{C}{**}), \mathrm{C}(\mathrm{C}{**})(\mathrm{C}{**}), \dots\}$ ("cototal").
- Locally correct, but possibly non well-founded derivations (Mints 1978).

Finitary algebras: consistency, entailment, ideals

For **D** (derivations):

- $\{\mathrm{C}0*, \mathrm{C}{*}0\}$ is "consistent", written $\mathrm{C}0* \uparrow \mathrm{C}{*}0$.
- $\{\mathrm{C}0*, \mathrm{C}{*}0\} \vdash \mathrm{C}00$ ("entails").
- Ideals: consistent and "deductively closed" sets of tokens.

Examples of ideals:

- $\{\mathrm{C}0*, \mathrm{C}{**}\}$.
- $\{\mathrm{C}00, \mathrm{C}0*, \mathrm{C}{*}0, \mathrm{C}{**}\}$.
- The deductive closure of a finite (well-founded) derivation.
- $\{\mathrm{C}{**}, \mathrm{C}(\mathrm{C}{**})*, \mathrm{C}{*}(\mathrm{C}{**}), \mathrm{C}(\mathrm{C}{**})(\mathrm{C}{**}), \dots\}$ ("cototal").
- Locally correct, but possibly non well-founded derivations (Mints 1978).

Finitary algebras: consistency, entailment, ideals

For **D** (derivations):

- $\{\mathrm{C}0*, \mathrm{C}{*}0\}$ is "consistent", written $\mathrm{C}0* \uparrow \mathrm{C}{*}0$.
- $\{\mathrm{C}0*, \mathrm{C}{*}0\} \vdash \mathrm{C}00$ ("entails").
- Ideals: consistent and "deductively closed" sets of tokens.

Examples of ideals:

- $\{\mathrm{C}0*, \mathrm{C}{*}{*}\}$.
- $\{\mathrm{C}00, \mathrm{C}0*, \mathrm{C}{*}0, \mathrm{C}{*}{*}\}$.
- The deductive closure of a finite (well-founded) derivation.
- $\{\mathrm{C}{*}{*}, \mathrm{C}(\mathrm{C}{*}{*}){*}, \mathrm{C}{*}(\mathrm{C}{*}{*}), \mathrm{C}(\mathrm{C}{*}{*})(\mathrm{C}{*}{*}), \dots\}$ ("cototal").
- Locally correct, but possibly non well-founded derivations (Mints 1978).

Finitary algebras: consistency, entailment, ideals

For **D** (derivations):

- $\{\mathrm{C}0*, \mathrm{C}*0\}$ is "consistent", written $\mathrm{C}0* \uparrow \mathrm{C}*0$.
- $\{\mathrm{C}0*, \mathrm{C}*0\} \vdash \mathrm{C}00$ ("entails").
- Ideals: consistent and "deductively closed" sets of tokens.

Examples of ideals:

- $\{\mathrm{C}0*, \mathrm{C}**\}$.
- $\{\mathrm{C}00, \mathrm{C}0*, \mathrm{C}*0, \mathrm{C}**\}$.
- The deductive closure of a finite (well-founded) derivation.
- $\{\mathrm{C}**, \mathrm{C}(\mathrm{C}**)*, \mathrm{C}*(\mathrm{C}**), \mathrm{C}(\mathrm{C}**)(\mathrm{C}**), \dots\}$ ("cototal").
- Locally correct, but possibly non well-founded derivations (Mints 1978).

Finitary algebras: consistency, entailment, ideals

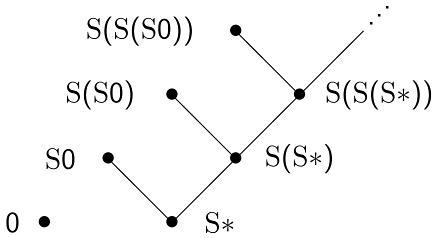
For **D** (derivations):

- $\{\mathrm{C}0*, \mathrm{C}{*}0\}$ is "consistent", written $\mathrm{C}0* \uparrow \mathrm{C}{*}0$.
- $\{\mathrm{C}0*, \mathrm{C}{*}0\} \vdash \mathrm{C}00$ ("entails").
- Ideals: consistent and "deductively closed" sets of tokens.

Examples of ideals:

- $\{\mathrm{C}0*, \mathrm{C}{*}{*}\}$.
- $\{\mathrm{C}00, \mathrm{C}0*, \mathrm{C}{*}0, \mathrm{C}{*}{*}\}$.
- The deductive closure of a finite (well-founded) derivation.
- $\{\mathrm{C}{*}{*}, \mathrm{C}(\mathrm{C}{*}{*})*, \mathrm{C}{*}(\mathrm{C}{*}{*}), \mathrm{C}(\mathrm{C}{*}{*})(\mathrm{C}{*}{*}), \dots\}$ ("cototal").
- Locally correct, but possibly non well-founded derivations (Mints 1978).

Tokens and entailment for **N**



Constructors as continuous functions

- Continuous maps $f\colon |\mathbf{N}| \to |\mathbf{N}|$ (see below) are monotone: $x \subseteq y \to fx \subseteq fy$.
- Easy: every constructor gives rise to a continuous function.
- Want: constructors have disjoint ranges and are injective (cf. the Peano axioms $\mathrm{S}x \neq 0$ and $\mathrm{S}x = \mathrm{S}y \to x = y$).
- This holds for non-flat algebras, but not for flat ones:

$$\begin{array}{cccc} 0 & \mathrm{S}0 & \mathrm{S}(\mathrm{S}0) & \\ \bullet & \bullet & \bullet & \cdots \end{array}$$

There constructors must be strict (i.e., $\mathrm{C}\vec{x}\emptyset\vec{y} = \emptyset$), hence

In **P**: $\mathrm{S}_1\emptyset = \emptyset = \mathrm{S}_2\emptyset$,

In **D**: $\mathrm{C}\emptyset\{0\} = \emptyset = \mathrm{C}\{0\}\emptyset$.

Constructors as continuous functions

- Continuous maps $f\colon |\mathbf{N}| \to |\mathbf{N}|$ (see below) are monotone: $x \subseteq y \to fx \subseteq fy$.
- Easy: every constructor gives rise to a continuous function.
- Want: constructors have disjoint ranges and are injective (cf. the Peano axioms $\mathrm{S}x \neq 0$ and $\mathrm{S}x = \mathrm{S}y \to x = y$).
- This holds for non-flat algebras, but not for flat ones:

$$\begin{array}{cccc} 0 & \mathrm{S}0 & \mathrm{S}(\mathrm{S}0) & \\ \bullet & \bullet & \bullet & \cdots \end{array}$$

There constructors must be strict (i.e., $\mathrm{C}\vec{x}\emptyset\vec{y} = \emptyset$), hence

In **P**: $\quad \mathrm{S}_1\emptyset = \emptyset = \mathrm{S}_2\emptyset,$

In **D**: $\quad \mathrm{C}\emptyset\{0\} = \emptyset = \mathrm{C}\{0\}\emptyset.$

Constructors as continuous functions

- Continuous maps $f\colon |\mathbf{N}| \to |\mathbf{N}|$ (see below) are monotone: $x \subseteq y \to fx \subseteq fy$.
- Easy: every constructor gives rise to a continuous function.
- Want: constructors have disjoint ranges and are injective (cf. the Peano axioms $\mathrm{S}x \neq 0$ and $\mathrm{S}x = \mathrm{S}y \to x = y$).
- This holds for non-flat algebras, but not for flat ones:

$$\begin{array}{cccc} 0 & \mathrm{S}0 & \mathrm{S}(\mathrm{S}0) & \\ \bullet & \bullet & \bullet & \dots \end{array}$$

There constructors must be strict (i.e., $\mathrm{C}\vec{x}\emptyset\vec{y} = \emptyset$), hence

In $\mathbf{P}$: $\quad \mathrm{S}_1\emptyset = \emptyset = \mathrm{S}_2\emptyset$,

In $\mathbf{D}$: $\quad \mathrm{C}\emptyset\{0\} = \emptyset = \mathrm{C}\{0\}\emptyset$.

Constructors as continuous functions

- Continuous maps $f\colon |\mathbf{N}| \to |\mathbf{N}|$ (see below) are monotone: $x \subseteq y \to fx \subseteq fy$.
- Easy: every constructor gives rise to a continuous function.
- Want: constructors have disjoint ranges and are injective (cf. the Peano axioms $\mathrm{S}x \neq 0$ and $\mathrm{S}x = \mathrm{S}y \to x = y$).
- This holds for non-flat algebras, but not for flat ones:

$$\begin{array}{cccc} 0 & \mathrm{S}0 & \mathrm{S}(\mathrm{S}0) & \\ \bullet & \bullet & \bullet & \dots \end{array}$$

There constructors must be strict (i.e., $\mathrm{C}\vec{x}\emptyset\vec{y} = \emptyset$), hence

$$\begin{aligned} &\text{In } \mathbf{P}\text{:} \quad \mathrm{S}_1\emptyset = \emptyset = \mathrm{S}_2\emptyset, \\ &\text{In } \mathbf{D}\text{:} \quad \mathrm{C}\emptyset\{0\} = \emptyset = \mathrm{C}\{0\}\emptyset. \end{aligned}$$

Constructors as continuous functions

- Continuous maps $f\colon |\mathbf{N}| \to |\mathbf{N}|$ (see below) are monotone: $x \subseteq y \to fx \subseteq fy$.
- Easy: every constructor gives rise to a continuous function.
- Want: constructors have disjoint ranges and are injective (cf. the Peano axioms $\mathrm{S}x \neq 0$ and $\mathrm{S}x = \mathrm{S}y \to x = y$).
- This holds for non-flat algebras, but not for flat ones:

$$\begin{array}{cccc} 0 & \mathrm{S}0 & \mathrm{S}(\mathrm{S}0) & \\ \bullet & \bullet & \bullet & \cdots \end{array}$$

There constructors must be strict (i.e., $\mathrm{C}\vec{x}\emptyset\vec{y} = \emptyset$), hence

$$\begin{aligned} \text{In } \mathbf{P}: &\quad \mathrm{S}_1\emptyset = \emptyset = \mathrm{S}_2\emptyset, \\ \text{In } \mathbf{D}: &\quad \mathrm{C}\emptyset\{0\} = \emptyset = \mathrm{C}\{0\}\emptyset. \end{aligned}$$

Types

- Every mathematical object has a type.
- Types: built from base types (i.e., algebras) by $\rho \to \sigma$, $\rho \times \sigma$.
- $\rho \times \sigma$ can be seen as finitary algebra with two parameters.
- Types and propositions are kept separate.
- Non-dependent types suffice.

Types

- Every mathematical object has a type.
- Types: built from base types (i.e., algebras) by $\rho \to \sigma$, $\rho \times \sigma$.
- $\rho \times \sigma$ can be seen as finitary algebra with two parameters.
- Types and propositions are kept separate.
- Non-dependent types suffice.

Types

- Every mathematical object has a type.
- Types: built from base types (i.e., algebras) by $\rho \to \sigma$, $\rho \times \sigma$.
- $\rho \times \sigma$ can be seen as finitary algebra with two parameters.
- Types and propositions are kept separate.
- Non-dependent types suffice.

Types

- Every mathematical object has a type.
- Types: built from base types (i.e., algebras) by $\rho \to \sigma$, $\rho \times \sigma$.
- $\rho \times \sigma$ can be seen as finitary algebra with two parameters.
- Types and propositions are kept separate.
- Non-dependent types suffice.

Types

- Every mathematical object has a type.
- Types: built from base types (i.e., algebras) by $\rho \to \sigma$, $\rho \times \sigma$.
- $\rho \times \sigma$ can be seen as finitary algebra with two parameters.
- Types and propositions are kept separate.
- Non-dependent types suffice.

The Scott-Ershov model of partial continuous functionals

- Let $\mathbf{A} = (A, \mathrm{Con}_A, \vdash_A)$, $\mathbf{B} = (B, \mathrm{Con}_B, \vdash_B)$ be information systems (Scott). Function space: $\mathbf{A} \to \mathbf{B} := (C, \mathrm{Con}, \vdash)$, with

$$C := \mathrm{Con}_A \times B,$$

$$\{(U_i, b_i)\}_{i \in I} \in \mathrm{Con} := \forall_{J \subseteq I} (\bigcup_{j \in J} U_j \in \mathrm{Con}_A \to \{b_j\}_{j \in J} \in \mathrm{Con}_B),$$

$$\{(U_i, b_i)\}_{i \in I} \vdash U := (\{\, b_i \mid U \vdash_A U_i \,\} \vdash_B U).$$

- Partial continuous functionals of type ρ: the ideals in $\mathbf{C}_\rho$.

$$\mathbf{C}_\iota := (\mathrm{Tok}_\iota, \mathrm{Con}_\iota, \vdash_\iota), \qquad \mathbf{C}_{\rho \to \sigma} := \mathbf{C}_\rho \to \mathbf{C}_\sigma.$$

- $f \in |\mathbf{C}_\rho|$: limit of formal neighborhoods $U \in \mathrm{Con}_{\rho \to \sigma}$.
- $f \in |\mathbf{C}_\rho|$ computable: r.e. limit.

The Scott-Ershov model of partial continuous functionals

- Let $\mathbf{A} = (A, \mathrm{Con}_A, \vdash_A)$, $\mathbf{B} = (B, \mathrm{Con}_B, \vdash_B)$ be information systems (Scott). Function space: $\mathbf{A} \to \mathbf{B} := (C, \mathrm{Con}, \vdash)$, with

$$C := \mathrm{Con}_A \times B,$$

$$\{(U_i, b_i)\}_{i \in I} \in \mathrm{Con} := \forall_{J \subseteq I}(\bigcup_{j \in J} U_j \in \mathrm{Con}_A \to \{b_j\}_{j \in J} \in \mathrm{Con}_B),$$

$$\{(U_i, b_i)\}_{i \in I} \vdash U := (\{\, b_i \mid U \vdash_A U_i \,\} \vdash_B U).$$

- Partial continuous functionals of type ρ: the ideals in $\mathbf{C}_\rho$.

$$\mathbf{C}_\iota := (\mathrm{Tok}_\iota, \mathrm{Con}_\iota, \vdash_\iota), \qquad \mathbf{C}_{\rho \to \sigma} := \mathbf{C}_\rho \to \mathbf{C}_\sigma.$$

- $f \in |\mathbf{C}_\rho|$: limit of formal neighborhoods $U \in \mathrm{Con}_{\rho \to \sigma}$.
- $f \in |\mathbf{C}_\rho|$ computable: r.e. limit.

The Scott-Ershov model of partial continuous functionals

- Let $\mathbf{A} = (A, \mathrm{Con}_A, \vdash_A)$, $\mathbf{B} = (B, \mathrm{Con}_B, \vdash_B)$ be information systems (Scott). Function space: $\mathbf{A} \to \mathbf{B} := (C, \mathrm{Con}, \vdash)$, with

$$C := \mathrm{Con}_A \times B,$$

$$\{(U_i, b_i)\}_{i \in I} \in \mathrm{Con} := \forall_{J \subseteq I} (\bigcup_{j \in J} U_j \in \mathrm{Con}_A \to \{b_j\}_{j \in J} \in \mathrm{Con}_B),$$

$$\{(U_i, b_i)\}_{i \in I} \vdash U := (\{\, b_i \mid U \vdash_A U_i \,\} \vdash_B U).$$

- Partial continuous functionals of type ρ: the ideals in $\mathbf{C}_\rho$.

$$\mathbf{C}_\iota := (\mathrm{Tok}_\iota, \mathrm{Con}_\iota, \vdash_\iota), \qquad \mathbf{C}_{\rho \to \sigma} := \mathbf{C}_\rho \to \mathbf{C}_\sigma.$$

- $f \in |\mathbf{C}_\rho|$: limit of formal neighborhoods $U \in \mathrm{Con}_{\rho \to \sigma}$.
- $f \in |\mathbf{C}_\rho|$ computable: r.e. limit.

The Scott-Ershov model of partial continuous functionals

- Let $\mathbf{A} = (A, \mathrm{Con}_A, \vdash_A)$, $\mathbf{B} = (B, \mathrm{Con}_B, \vdash_B)$ be information systems (Scott). Function space: $\mathbf{A} \to \mathbf{B} := (C, \mathrm{Con}, \vdash)$, with

$$C := \mathrm{Con}_A \times B,$$

$$\{(U_i, b_i)\}_{i \in I} \in \mathrm{Con} := \forall_{J \subseteq I}(\bigcup_{j \in J} U_j \in \mathrm{Con}_A \to \{b_j\}_{j \in J} \in \mathrm{Con}_B),$$

$$\{(U_i, b_i)\}_{i \in I} \vdash U := (\{\, b_i \mid U \vdash_A U_i \,\} \vdash_B U).$$

- Partial continuous functionals of type ρ: the ideals in $\mathbf{C}_\rho$.

$$\mathbf{C}_\iota := (\mathrm{Tok}_\iota, \mathrm{Con}_\iota, \vdash_\iota), \qquad \mathbf{C}_{\rho \to \sigma} := \mathbf{C}_\rho \to \mathbf{C}_\sigma.$$

- $f \in |\mathbf{C}_\rho|$: limit of formal neighborhoods $U \in \mathrm{Con}_{\rho \to \sigma}$.
- $f \in |\mathbf{C}_\rho|$ computable: r.e. limit.

The Scott-Ershov model of partial continuous functionals

- Let $\mathbf{A} = (A, \mathrm{Con}_A, \vdash_A)$, $\mathbf{B} = (B, \mathrm{Con}_B, \vdash_B)$ be information systems (Scott). Function space: $\mathbf{A} \to \mathbf{B} := (C, \mathrm{Con}, \vdash)$, with

$$C := \mathrm{Con}_A \times B,$$

$$\{(U_i, b_i)\}_{i \in I} \in \mathrm{Con} := \forall_{J \subseteq I}(\bigcup_{j \in J} U_j \in \mathrm{Con}_A \to \{b_j\}_{j \in J} \in \mathrm{Con}_B),$$

$$\{(U_i, b_i)\}_{i \in I} \vdash U := (\{\, b_i \mid U \vdash_A U_i \,\} \vdash_B U).$$

- Partial continuous functionals of type ρ: the ideals in $\mathbf{C}_\rho$.

$$\mathbf{C}_\iota := (\mathrm{Tok}_\iota, \mathrm{Con}_\iota, \vdash_\iota), \qquad \mathbf{C}_{\rho \to \sigma} := \mathbf{C}_\rho \to \mathbf{C}_\sigma.$$

- $f \in |\mathbf{C}_\rho|$: limit of formal neighborhoods $U \in \mathrm{Con}_{\rho \to \sigma}$.
- $f \in |\mathbf{C}_\rho|$ computable: r.e. limit.

The Scott-Ershov model of partial continuous functionals

- Let $\mathbf{A} = (A, \mathrm{Con}_A, \vdash_A)$, $\mathbf{B} = (B, \mathrm{Con}_B, \vdash_B)$ be information systems (Scott). Function space: $\mathbf{A} \to \mathbf{B} := (C, \mathrm{Con}, \vdash)$, with

$$C := \mathrm{Con}_A \times B,$$

$$\{(U_i, b_i)\}_{i \in I} \in \mathrm{Con} := \forall_{J \subseteq I}(\bigcup_{j \in J} U_j \in \mathrm{Con}_A \to \{b_j\}_{j \in J} \in \mathrm{Con}_B),$$

$$\{(U_i, b_i)\}_{i \in I} \vdash U := (\{\, b_i \mid U \vdash_A U_i \,\} \vdash_B U).$$

- Partial continuous functionals of type ρ: the ideals in $\mathbf{C}_\rho$.

$$\mathbf{C}_\iota := (\mathrm{Tok}_\iota, \mathrm{Con}_\iota, \vdash_\iota), \qquad \mathbf{C}_{\rho \to \sigma} := \mathbf{C}_\rho \to \mathbf{C}_\sigma.$$

- $f \in |\mathbf{C}_\rho|$: limit of formal neighborhoods $U \in \mathrm{Con}_{\rho \to \sigma}$.
- $f \in |\mathbf{C}_\rho|$ computable: r.e. limit.

The Scott-Ershov model of partial continuous functionals

- Let $\mathbf{A} = (A, \mathrm{Con}_A, \vdash_A)$, $\mathbf{B} = (B, \mathrm{Con}_B, \vdash_B)$ be information systems (Scott). Function space: $\mathbf{A} \to \mathbf{B} := (C, \mathrm{Con}, \vdash)$, with

$$C := \mathrm{Con}_A \times B,$$

$$\{(U_i, b_i)\}_{i \in I} \in \mathrm{Con} := \forall_{J \subseteq I}(\bigcup_{j \in J} U_j \in \mathrm{Con}_A \to \{b_j\}_{j \in J} \in \mathrm{Con}_B),$$

$$\{(U_i, b_i)\}_{i \in I} \vdash U := (\{\, b_i \mid U \vdash_A U_i \,\} \vdash_B U).$$

- Partial continuous functionals of type ρ: the ideals in $\mathbf{C}_\rho$.

$$\mathbf{C}_\iota := (\mathrm{Tok}_\iota, \mathrm{Con}_\iota, \vdash_\iota), \qquad \mathbf{C}_{\rho \to \sigma} := \mathbf{C}_\rho \to \mathbf{C}_\sigma.$$

- $f \in |\mathbf{C}_\rho|$: limit of formal neighborhoods $U \in \mathrm{Con}_{\rho \to \sigma}$.
- $f \in |\mathbf{C}_\rho|$ computable: r.e. limit.

Terms

- Terms are built from (typed) variables and (typed) constants (constructors C or defined constants D, see below):

$$M, N ::= x^\rho \mid \mathrm{C}^\rho \mid D^\rho \mid (\lambda_{x^\rho} M^\sigma)^{\rho\to\sigma} \mid (M^{\rho\to\sigma} N^\rho)^\sigma.$$

- Every defined constant D comes with a system of computation rules $D\vec{P}_i(\vec{y}_i) = M_i$ with $\mathrm{FV}(M_i) \subseteq \vec{y}_i$.
- $\vec{P}_i(\vec{y}_i)$: "constructor patterns", i.e., lists of applicative terms built from constructors and distinct variables, with each constructor C occurring in a context $\mathrm{C}\vec{P}$ (of base type). We assume that $\vec{P}_i$ and $\vec{P}_j$ for $i \neq j$ are non-unifiable.

Examples:

- Predecessor $\mathrm{P} \colon \mathbf{N} \to \mathbf{N}$, defined by $\mathrm{P}0 = 0$, $\mathrm{P}(\mathrm{S}n) = n$,
- Gödel's primitive recursion operators $\mathcal{R}_{\mathbf{N}}^\tau \colon \mathbf{N} \to \tau \to (\mathbf{N} \to \tau \to \tau) \to \tau$ with computation rules $\mathcal{R}0fg = f$, $\mathcal{R}(\mathrm{S}n)fg = gn(\mathcal{R}nfg)$, and
- the least-fixed-point operators Y_ρ of type $(\rho \to \rho) \to \rho$ defined by the computation rule $Y_\rho f = f(Y_\rho f)$.

Terms

- Terms are built from (typed) variables and (typed) constants (constructors C or defined constants D, see below):

$$M, N ::= x^\rho \mid \mathrm{C}^\rho \mid D^\rho \mid (\lambda_{x^\rho} M^\sigma)^{\rho \to \sigma} \mid (M^{\rho \to \sigma} N^\rho)^\sigma.$$

- Every defined constant D comes with a system of computation rules $D\vec{P}_i(\vec{y}_i) = M_i$ with $\mathrm{FV}(M_i) \subseteq \vec{y}_i$.
- $\vec{P}_i(\vec{y}_i)$: "constructor patterns", i.e., lists of applicative terms built from constructors and distinct variables, with each constructor C occurring in a context $\mathrm{C}\vec{P}$ (of base type). We assume that $\vec{P}_i$ and $\vec{P}_j$ for $i \neq j$ are non-unifiable.

Examples:

- Predecessor $\mathrm{P}\colon \mathbf{N} \to \mathbf{N}$, defined by $\mathrm{P}0 = 0$, $\mathrm{P}(\mathrm{S}n) = n$,
- Gödel's primitive recursion operators $\mathcal{R}_{\mathbf{N}}^\tau\colon \mathbf{N} \to \tau \to (\mathbf{N} \to \tau \to \tau) \to \tau$ with computation rules $\mathcal{R}0fg = f$, $\mathcal{R}(\mathrm{S}n)fg = gn(\mathcal{R}nfg)$, and
- the least-fixed-point operators Y_ρ of type $(\rho \to \rho) \to \rho$ defined by the computation rule $Y_\rho f = f(Y_\rho f)$.

Terms

- Terms are built from (typed) variables and (typed) constants (constructors C or defined constants D, see below):

$$M, N ::= x^\rho \mid \mathrm{C}^\rho \mid D^\rho \mid (\lambda_{x^\rho} M^\sigma)^{\rho \to \sigma} \mid (M^{\rho \to \sigma} N^\rho)^\sigma.$$

- Every defined constant D comes with a system of computation rules $D\vec{P}_i(\vec{y}_i) = M_i$ with $\mathrm{FV}(M_i) \subseteq \vec{y}_i$.
- $\vec{P}_i(\vec{y}_i)$: "constructor patterns", i.e., lists of applicative terms built from constructors and distinct variables, with each constructor C occurring in a context $\mathrm{C}\vec{P}$ (of base type). We assume that $\vec{P}_i$ and $\vec{P}_j$ for $i \neq j$ are non-unifiable.

Examples:

- Predecessor $\mathrm{P} \colon \mathbf{N} \to \mathbf{N}$, defined by $\mathrm{P}0 = 0$, $\mathrm{P}(\mathrm{S}n) = n$,
- Gödel's primitive recursion operators $\mathcal{R}^\tau_{\mathbf{N}} \colon \mathbf{N} \to \tau \to (\mathbf{N} \to \tau \to \tau) \to \tau$ with computation rules $\mathcal{R}0fg = f$, $\mathcal{R}(\mathrm{S}n)fg = gn(\mathcal{R}nfg)$, and
- the least-fixed-point operators Y_ρ of type $(\rho \to \rho) \to \rho$ defined by the computation rule $Y_\rho f = f(Y_\rho f)$.

Terms

- Terms are built from (typed) variables and (typed) constants (constructors C or defined constants D, see below):
$$M, N ::= x^\rho \mid \mathrm{C}^\rho \mid D^\rho \mid (\lambda_{x^\rho} M^\sigma)^{\rho\to\sigma} \mid (M^{\rho\to\sigma} N^\rho)^\sigma.$$
- Every defined constant D comes with a system of computation rules $D\vec{P}_i(\vec{y}_i) = M_i$ with $\mathrm{FV}(M_i) \subseteq \vec{y}_i$.
- $\vec{P}_i(\vec{y}_i)$: "constructor patterns", i.e., lists of applicative terms built from constructors and distinct variables, with each constructor C occurring in a context $\mathrm{C}\vec{P}$ (of base type). We assume that $\vec{P}_i$ and $\vec{P}_j$ for $i \neq j$ are non-unifiable.

Examples:

- Predecessor $\mathrm{P}\colon \mathbf{N} \to \mathbf{N}$, defined by $\mathrm{P}0 = 0$, $\mathrm{P}(\mathrm{S}n) = n$,
- Gödel's primitive recursion operators $\mathcal{R}_{\mathbf{N}}^{\tau}\colon \mathbf{N} \to \tau \to (\mathbf{N} \to \tau \to \tau) \to \tau$ with computation rules $\mathcal{R}0fg = f$, $\mathcal{R}(\mathrm{S}n)fg = gn(\mathcal{R}nfg)$, and
- the least-fixed-point operators Y_ρ of type $(\rho \to \rho) \to \rho$ defined by the computation rule $Y_\rho f = f(Y_\rho f)$.

Terms

- Terms are built from (typed) variables and (typed) constants (constructors C or defined constants D, see below):

$$M, N ::= x^\rho \mid \mathrm{C}^\rho \mid D^\rho \mid (\lambda_{x^\rho} M^\sigma)^{\rho\to\sigma} \mid (M^{\rho\to\sigma} N^\rho)^\sigma.$$

- Every defined constant D comes with a system of computation rules $D\vec{P}_i(\vec{y}_i) = M_i$ with $\mathrm{FV}(M_i) \subseteq \vec{y}_i$.
- $\vec{P}_i(\vec{y}_i)$: "constructor patterns", i.e., lists of applicative terms built from constructors and distinct variables, with each constructor C occurring in a context $\mathrm{C}\vec{P}$ (of base type). We assume that $\vec{P}_i$ and $\vec{P}_j$ for $i \neq j$ are non-unifiable.

Examples:

- Predecessor $\mathrm{P}\colon \mathbf{N} \to \mathbf{N}$, defined by $\mathrm{P}0 = 0$, $\mathrm{P}(\mathrm{S}n) = n$,
- Gödel's primitive recursion operators $\mathcal{R}^\tau_{\mathbf{N}}\colon \mathbf{N} \to \tau \to (\mathbf{N} \to \tau \to \tau) \to \tau$ with computation rules $\mathcal{R}0fg = f$, $\mathcal{R}(\mathrm{S}n)fg = gn(\mathcal{R}nfg)$, and
- the least-fixed-point operators Y_ρ of type $(\rho \to \rho) \to \rho$ defined by the computation rule $Y_\rho f = f(Y_\rho f)$.

Terms

- Terms are built from (typed) variables and (typed) constants (constructors C or defined constants D, see below):

$$M, N ::= x^\rho \mid \mathrm{C}^\rho \mid D^\rho \mid (\lambda_{x^\rho} M^\sigma)^{\rho \to \sigma} \mid (M^{\rho \to \sigma} N^\rho)^\sigma.$$

- Every defined constant D comes with a system of computation rules $D\vec{P}_i(\vec{y}_i) = M_i$ with $\mathrm{FV}(M_i) \subseteq \vec{y}_i$.
- $\vec{P}_i(\vec{y}_i)$: "constructor patterns", i.e., lists of applicative terms built from constructors and distinct variables, with each constructor C occurring in a context $\mathrm{C}\vec{P}$ (of base type). We assume that $\vec{P}_i$ and $\vec{P}_j$ for $i \neq j$ are non-unifiable.

Examples:

- Predecessor $\mathrm{P} \colon \mathbf{N} \to \mathbf{N}$, defined by $\mathrm{P}0 = 0$, $\mathrm{P}(\mathrm{S}n) = n$,
- Gödel's primitive recursion operators $\mathcal{R}^\tau_{\mathbf{N}} \colon \mathbf{N} \to \tau \to (\mathbf{N} \to \tau \to \tau) \to \tau$ with computation rules $\mathcal{R}0fg = f$, $\mathcal{R}(\mathrm{S}n)fg = gn(\mathcal{R}nfg)$, and
- the least-fixed-point operators Y_ρ of type $(\rho \to \rho) \to \rho$ defined by the computation rule $Y_\rho f = f(Y_\rho f)$.

Denotational semantics

For every closed term $\lambda_{\vec{x}} M$ of type $\vec{\rho} \to \sigma$ we inductively define a set $[\![\lambda_{\vec{x}} M]\!]$ of tokens of type $\vec{\rho} \to \sigma$.

$$\frac{U_i \vdash b}{(\vec{U}, b) \in [\![\lambda_{\vec{x}} x_i]\!]}(V), \qquad \frac{(\vec{U}, V, c) \in [\![\lambda_{\vec{x}} M]\!] \qquad (\vec{U}, V) \subseteq [\![\lambda_{\vec{x}} N]\!]}{(\vec{U}, c) \in [\![\lambda_{\vec{x}} (MN)]\!]}(A).$$

For every constructor C and defined constant D:

$$\frac{\vec{V} \vdash \vec{b^*}}{(\vec{U}, \vec{V}, \mathrm{C}\vec{b^*}) \in [\![\lambda_{\vec{x}} \mathrm{C}]\!]}(\mathrm{C}), \qquad \frac{(\vec{U}, \vec{V}, b) \in [\![\lambda_{\vec{x},\vec{y}} M]\!] \qquad \vec{W} \vdash \vec{P}(\vec{V})}{(\vec{U}, \vec{W}, b) \in [\![\lambda_{\vec{x}} D]\!]}(D),$$

with one rule (D) for every computation rule $D\vec{P}(\vec{y}) = M$. Note:

$(\vec{U}, b)$ denotes $(U_1, \dots (U_n, b) \dots)$,

$(\vec{U}, V) \subseteq [\![\lambda_{\vec{x}} M]\!]$ means $(\vec{U}, b) \in [\![\lambda_{\vec{x}} M]\!]$ for all $b \in V$.

Denotational semantics

For every closed term $\lambda_{\vec{x}} M$ of type $\vec{\rho} \to \sigma$ we inductively define a set $[\![\lambda_{\vec{x}} M]\!]$ of tokens of type $\vec{\rho} \to \sigma$.

$$\frac{U_i \vdash b}{(\vec{U}, b) \in [\![\lambda_{\vec{x}} x_i]\!]}(V), \qquad \frac{(\vec{U}, V, c) \in [\![\lambda_{\vec{x}} M]\!] \qquad (\vec{U}, V) \subseteq [\![\lambda_{\vec{x}} N]\!]}{(\vec{U}, c) \in [\![\lambda_{\vec{x}}(MN)]\!]}(A).$$

For every constructor C and defined constant D:

$$\frac{\vec{V} \vdash \vec{b^*}}{(\vec{U}, \vec{V}, \mathrm{C}\vec{b^*}) \in [\![\lambda_{\vec{x}} \mathrm{C}]\!]}(\mathrm{C}), \qquad \frac{(\vec{U}, \vec{V}, b) \in [\![\lambda_{\vec{x},\vec{y}} M]\!] \qquad \vec{W} \vdash \vec{P}(\vec{V})}{(\vec{U}, \vec{W}, b) \in [\![\lambda_{\vec{x}} D]\!]}(D),$$

with one rule (D) for every computation rule $D\vec{P}(\vec{y}) = M$. Note:

$(\vec{U}, b)$ denotes $(U_1, \dots (U_n, b) \dots)$,

$(\vec{U}, V) \subseteq [\![\lambda_{\vec{x}} M]\!]$ means $(\vec{U}, b) \in [\![\lambda_{\vec{x}} M]\!]$ for all $b \in V$.

Denotational semantics

For every closed term $\lambda_{\vec{x}} M$ of type $\vec{\rho} \to \sigma$ we inductively define a set $[\![\lambda_{\vec{x}} M]\!]$ of tokens of type $\vec{\rho} \to \sigma$.

$$\frac{U_i \vdash b}{(\vec{U}, b) \in [\![\lambda_{\vec{x}} x_i]\!]}(V), \qquad \frac{(\vec{U}, V, c) \in [\![\lambda_{\vec{x}} M]\!] \qquad (\vec{U}, V) \subseteq [\![\lambda_{\vec{x}} N]\!]}{(\vec{U}, c) \in [\![\lambda_{\vec{x}} (MN)]\!]}(A).$$

For every constructor C and defined constant D:

$$\frac{\vec{V} \vdash \vec{b^*}}{(\vec{U}, \vec{V}, \mathrm{C}\vec{b^*}) \in [\![\lambda_{\vec{x}} \mathrm{C}]\!]}(\mathrm{C}), \qquad \frac{(\vec{U}, \vec{V}, b) \in [\![\lambda_{\vec{x},\vec{y}} M]\!] \qquad \vec{W} \vdash \vec{P}(\vec{V})}{(\vec{U}, \vec{W}, b) \in [\![\lambda_{\vec{x}} D]\!]}(D),$$

with one rule (D) for every computation rule $D\vec{P}(\vec{y}\,) = M$. Note:

$(\vec{U}, b)$ denotes $(U_1, \dots (U_n, b) \dots)$,

$(\vec{U}, V) \subseteq [\![\lambda_{\vec{x}} M]\!]$ means $(\vec{U}, b) \in [\![\lambda_{\vec{x}} M]\!]$ for all $b \in V$.

Denotational semantics

For every closed term $\lambda_{\vec{x}} M$ of type $\vec{\rho} \to \sigma$ we inductively define a set $[\![\lambda_{\vec{x}} M]\!]$ of tokens of type $\vec{\rho} \to \sigma$.

$$\frac{U_i \vdash b}{(\vec{U}, b) \in [\![\lambda_{\vec{x}} x_i]\!]}(V), \qquad \frac{(\vec{U}, V, c) \in [\![\lambda_{\vec{x}} M]\!] \qquad (\vec{U}, V) \subseteq [\![\lambda_{\vec{x}} N]\!]}{(\vec{U}, c) \in [\![\lambda_{\vec{x}} (MN)]\!]}(A).$$

For every constructor C and defined constant D:

$$\frac{\vec{V} \vdash \vec{b^*}}{(\vec{U}, \vec{V}, \mathrm{C}\vec{b^*}) \in [\![\lambda_{\vec{x}} \mathrm{C}]\!]}(\mathrm{C}), \qquad \frac{(\vec{U}, \vec{V}, b) \in [\![\lambda_{\vec{x}, \vec{y}} M]\!] \qquad \vec{W} \vdash \vec{P}(\vec{V})}{(\vec{U}, \vec{W}, b) \in [\![\lambda_{\vec{x}} D]\!]}(D),$$

with one rule (D) for every computation rule $D\vec{P}(\vec{y}\,) = M$. Note:

$(\vec{U}, b)$ denotes $(U_1, \dots (U_n, b) \dots)$,

$(\vec{U}, V) \subseteq [\![\lambda_{\vec{x}} M]\!]$ means $(\vec{U}, b) \in [\![\lambda_{\vec{x}} M]\!]$ for all $b \in V$.

Denotational semantics

For every closed term $\lambda_{\vec{x}} M$ of type $\vec{\rho} \to \sigma$ we inductively define a set $[\![\lambda_{\vec{x}} M]\!]$ of tokens of type $\vec{\rho} \to \sigma$.

$$\frac{U_i \vdash b}{(\vec{U}, b) \in [\![\lambda_{\vec{x}} x_i]\!]}(V), \qquad \frac{(\vec{U}, V, c) \in [\![\lambda_{\vec{x}} M]\!] \qquad (\vec{U}, V) \subseteq [\![\lambda_{\vec{x}} N]\!]}{(\vec{U}, c) \in [\![\lambda_{\vec{x}} (MN)]\!]}(A).$$

For every constructor C and defined constant D:

$$\frac{\vec{V} \vdash \vec{b^*}}{(\vec{U}, \vec{V}, \mathrm{C}\vec{b^*}) \in [\![\lambda_{\vec{x}} \mathrm{C}]\!]}(\mathrm{C}), \qquad \frac{(\vec{U}, \vec{V}, b) \in [\![\lambda_{\vec{x},\vec{y}} M]\!] \qquad \vec{W} \vdash \vec{P}(\vec{V})}{(\vec{U}, \vec{W}, b) \in [\![\lambda_{\vec{x}} D]\!]}(D),$$

with one rule (D) for every computation rule $D\vec{P}(\vec{y}\,) = M$. Note:

$(\vec{U}, b)$ denotes $(U_1, \dots (U_n, b) \dots)$,

$(\vec{U}, V) \subseteq [\![\lambda_{\vec{x}} M]\!]$ means $(\vec{U}, b) \in [\![\lambda_{\vec{x}} M]\!]$ for all $b \in V$.

Theorem

- *For every term M, $[\![\lambda_{\vec{x}} M]\!]$ is an ideal.*
- *If a term M converts to M' by $\beta\eta$-conversion or application of a computation rule, then $[\![M]\!] = [\![M']\!]$.*

Let

$$[\![M]\!]^{\vec{u}}_{\vec{x}} := \bigcup_{\vec{U} \subseteq \vec{u}} [\![M]\!]^{\vec{U}}_{\vec{x}} \quad \text{with} \quad [\![M]\!]^{\vec{U}}_{\vec{x}} := \{\, b \mid (\vec{U}, b) \in [\![\lambda_{\vec{x}} M]\!] \,\}.$$

A consequence of (A) is continuity of application:

$$c \in [\![MN]\!]^{\vec{u}}_{\vec{x}} \leftrightarrow \exists_{V \subseteq [\![N]\!]^{\vec{u}}_{\vec{x}}} ((V, c) \in [\![M]\!]^{\vec{u}}_{\vec{x}}).$$

Theorem

- *For every term M, $[\![\lambda_{\vec{x}} M]\!]$ is an ideal.*
- *If a term M converts to M' by $\beta\eta$-conversion or application of a computation rule, then $[\![M]\!] = [\![M']\!]$.*

Let

$$[\![M]\!]_{\vec{x}}^{\vec{u}} := \bigcup_{\vec{U} \subseteq \vec{u}} [\![M]\!]_{\vec{x}}^{\vec{U}} \quad \text{with} \quad [\![M]\!]_{\vec{x}}^{\vec{U}} := \{\, b \mid (\vec{U}, b) \in [\![\lambda_{\vec{x}} M]\!] \,\}.$$

A consequence of (A) is continuity of application:

$$c \in [\![MN]\!]_{\vec{x}}^{\vec{u}} \leftrightarrow \exists_{V \subseteq [\![N]\!]_{\vec{x}}^{\vec{u}}} ((V, c) \in [\![M]\!]_{\vec{x}}^{\vec{u}}).$$

Theorem

- *For every term M, $[\![\lambda_{\vec{x}} M]\!]$ is an ideal.*
- *If a term M converts to M' by $\beta\eta$-conversion or application of a computation rule, then $[\![M]\!] = [\![M']\!]$.*

Let

$$[\![M]\!]_{\vec{x}}^{\vec{u}} := \bigcup_{\vec{U} \subseteq \vec{u}} [\![M]\!]_{\vec{x}}^{\vec{U}} \quad \text{with} \quad [\![M]\!]_{\vec{x}}^{\vec{U}} := \{\, b \mid (\vec{U}, b) \in [\![\lambda_{\vec{x}} M]\!] \,\}.$$

A consequence of (A) is continuity of application:

$$c \in [\![MN]\!]_{\vec{x}}^{\vec{u}} \leftrightarrow \exists_{V \subseteq [\![N]\!]_{\vec{x}}^{\vec{u}}} ((V, c) \in [\![M]\!]_{\vec{x}}^{\vec{u}}).$$

Theorem

- *For every term M, $[\![\lambda_{\vec{x}} M]\!]$ is an ideal.*
- *If a term M converts to M' by $\beta\eta$-conversion or application of a computation rule, then $[\![M]\!] = [\![M']\!]$.*

Let

$$[\![M]\!]_{\vec{x}}^{\vec{u}} := \bigcup_{\vec{U} \subseteq \vec{u}} [\![M]\!]_{\vec{x}}^{\vec{U}} \quad \text{with} \quad [\![M]\!]_{\vec{x}}^{\vec{U}} := \{\, b \mid (\vec{U}, b) \in [\![\lambda_{\vec{x}} M]\!] \,\}.$$

A consequence of (A) is continuity of application:

$$c \in [\![MN]\!]_{\vec{x}}^{\vec{u}} \leftrightarrow \exists_{V \subseteq [\![N]\!]_{\vec{x}}^{\vec{u}}} ((V, c) \in [\![M]\!]_{\vec{x}}^{\vec{u}}).$$

Total functionals

The total ideals x of type ρ (notation $x \in G_\rho$) and an equivalence relation $x_1 \approx x_2$ between them are defined inductively.

- For an algebra ι, the total ideals x are those of the form $\mathrm{C}\vec{z}$ with C a constructor of ι and $\vec{z}$ total.
- $x_1 \approx_\iota x_2$ iff both are of the form $\mathrm{C}\vec{z}_i$ with the same constructor C of ι, and $z_{1j} \approx_\iota z_{2j}$ for all j.
- $f \in G_{\rho\to\sigma}$ iff $\forall_{z \in G_\rho}(fz \in G_\sigma)$.
- For $f, g \in G_{\rho\to\sigma}$ define $f \approx_{\rho\to\sigma} g$ by $\forall_{x \in G_\rho}(fx \approx_\sigma gx)$.

Theorem (Ershov 1974, Longo & Moggi 1984)

$x \approx_\rho y$ implies $fx \approx_\sigma fy$, for $x, y \in G_\rho$ and $f \in G_{\rho\to\sigma}$.

Total functionals

The total ideals x of type ρ (notation $x \in G_\rho$) and an equivalence relation $x_1 \approx x_2$ between them are defined inductively.

- For an algebra ι, the total ideals x are those of the form $\mathrm{C}\vec{z}$ with C a constructor of ι and $\vec{z}$ total.
- $x_1 \approx_\iota x_2$ iff both are of the form $\mathrm{C}\vec{z}_i$ with the same constructor C of ι, and $z_{1j} \approx_\iota z_{2j}$ for all j.
- $f \in G_{\rho\to\sigma}$ iff $\forall_{z \in G_\rho}(fz \in G_\sigma)$.
- For $f, g \in G_{\rho\to\sigma}$ define $f \approx_{\rho\to\sigma} g$ by $\forall_{x \in G_\rho}(fx \approx_\sigma gx)$.

Theorem (Ershov 1974, Longo & Moggi 1984)

$x \approx_\rho y$ implies $fx \approx_\sigma fy$, for $x, y \in G_\rho$ and $f \in G_{\rho\to\sigma}$.

Total functionals

The total ideals x of type ρ (notation $x \in G_\rho$) and an equivalence relation $x_1 \approx x_2$ between them are defined inductively.

- For an algebra ι, the total ideals x are those of the form $\mathrm{C}\vec{z}$ with C a constructor of ι and $\vec{z}$ total.
- $x_1 \approx_\iota x_2$ iff both are of the form $\mathrm{C}\vec{z}_i$ with the same constructor C of ι, and $z_{1j} \approx_\iota z_{2j}$ for all j.
- $f \in G_{\rho\to\sigma}$ iff $\forall_{z\in G_\rho}(fz \in G_\sigma)$.
- For $f, g \in G_{\rho\to\sigma}$ define $f \approx_{\rho\to\sigma} g$ by $\forall_{x\in G_\rho}(fx \approx_\sigma gx)$.

Theorem (Ershov 1974, Longo & Moggi 1984)

$x \approx_\rho y$ implies $fx \approx_\sigma fy$, for $x, y \in G_\rho$ and $f \in G_{\rho\to\sigma}$.

Total functionals

The total ideals x of type ρ (notation $x \in G_\rho$) and an equivalence relation $x_1 \approx x_2$ between them are defined inductively.

- For an algebra ι, the total ideals x are those of the form $\mathrm{C}\vec{z}$ with C a constructor of ι and $\vec{z}$ total.
- $x_1 \approx_\iota x_2$ iff both are of the form $\mathrm{C}\vec{z}_i$ with the same constructor C of ι, and $z_{1j} \approx_\iota z_{2j}$ for all j.
- $f \in G_{\rho\to\sigma}$ iff $\forall_{z \in G_\rho}(fz \in G_\sigma)$.
- For $f, g \in G_{\rho\to\sigma}$ define $f \approx_{\rho\to\sigma} g$ by $\forall_{x\in G_\rho}(fx \approx_\sigma gx)$.

Theorem (Ershov 1974, Longo & Moggi 1984)

$x \approx_\rho y$ implies $fx \approx_\sigma fy$, for $x, y \in G_\rho$ and $f \in G_{\rho\to\sigma}$.

Total functionals

The total ideals x of type ρ (notation $x \in G_\rho$) and an equivalence relation $x_1 \approx x_2$ between them are defined inductively.

- For an algebra ι, the total ideals x are those of the form $\mathrm{C}\vec{z}$ with C a constructor of ι and $\vec{z}$ total.
- $x_1 \approx_\iota x_2$ iff both are of the form $\mathrm{C}\vec{z}_i$ with the same constructor C of ι, and $z_{1j} \approx_\iota z_{2j}$ for all j.
- $f \in G_{\rho\to\sigma}$ iff $\forall_{z \in G_\rho}(fz \in G_\sigma)$.
- For $f, g \in G_{\rho\to\sigma}$ define $f \approx_{\rho\to\sigma} g$ by $\forall_{x\in G_\rho}(fx \approx_\sigma gx)$.

Theorem (Ershov 1974, Longo & Moggi 1984)

$x \approx_\rho y$ implies $fx \approx_\sigma fy$, for $x, y \in G_\rho$ and $f \in G_{\rho\to\sigma}$.

Total functionals

The total ideals x of type ρ (notation $x \in G_\rho$) and an equivalence relation $x_1 \approx x_2$ between them are defined inductively.

- For an algebra ι, the total ideals x are those of the form $\mathrm{C}\vec{z}$ with C a constructor of ι and $\vec{z}$ total.
- $x_1 \approx_\iota x_2$ iff both are of the form $\mathrm{C}\vec{z}_i$ with the same constructor C of ι, and $z_{1j} \approx_\iota z_{2j}$ for all j.
- $f \in G_{\rho\to\sigma}$ iff $\forall_{z \in G_\rho}(fz \in G_\sigma)$.
- For $f, g \in G_{\rho\to\sigma}$ define $f \approx_{\rho\to\sigma} g$ by $\forall_{x \in G_\rho}(fx \approx_\sigma gx)$.

Theorem (Ershov 1974, Longo & Moggi 1984)

$x \approx_\rho y$ implies $fx \approx_\sigma fy$, for $x, y \in G_\rho$ and $f \in G_{\rho\to\sigma}$.

Density

The total functionals are dense (w.r.t. the Scott topology) in the space of all partial continuous functionals of type ρ.

Theorem (Kreisel 1959, Ershov 1974, U. Berger 1993)

For every type $\rho = \rho_1 \to \ldots \to \rho_p \to \iota$ we have decidable formulas TExt_ρ and Sep^i_ρ ($i = 1, \ldots, p$) such that

- $\forall_{U \in \mathrm{Con}_\rho}(U \subseteq \{\, a \mid \mathrm{TExt}_\rho(U, a) \,\} \in G_\rho)$ *and*
- $\forall_{U,V \in \mathrm{Con}_\rho}(U \not\uparrow_\rho V \to \vec{z}_{U,V} \in G \wedge U\vec{z}_{U,V} \not\uparrow_\iota V\vec{z}_{U,V})$,

where $\vec{z}_{U,V} = z_{U,V,1}, \ldots, z_{U,V,p}$ and $z_{U,V,i} = \{\, a \mid \mathrm{Sep}^i_\rho(U, V, a) \,\}$.

Proof.
By induction on ρ. $\square$

Density

The total functionals are dense (w.r.t. the Scott topology) in the space of all partial continuous functionals of type ρ.

Theorem (Kreisel 1959, Ershov 1974, U. Berger 1993)

For every type $\rho = \rho_1 \to \ldots \to \rho_p \to \iota$ we have decidable formulas TExt_ρ and Sep^i_ρ $(i = 1, \ldots, p)$ such that

- $\forall_{U \in \mathrm{Con}_\rho}(U \subseteq \{\, a \mid \mathrm{TExt}_\rho(U, a) \,\} \in G_\rho)$ *and*
- $\forall_{U,V \in \mathrm{Con}_\rho}(U \not\asymp_\rho V \to \vec{z}_{U,V} \in G \wedge U\vec{z}_{U,V} \not\asymp_\iota V\vec{z}_{U,V})$,

where $\vec{z}_{U,V} = z_{U,V,1}, \ldots, z_{U,V,p}$ and $z_{U,V,i} = \{\, a \mid \mathrm{Sep}^i_\rho(U, V, a) \,\}$.

Proof.
By induction on ρ. $\square$

Density

The total functionals are dense (w.r.t. the Scott topology) in the space of all partial continuous functionals of type ρ.

Theorem (Kreisel 1959, Ershov 1974, U. Berger 1993)

For every type $\rho = \rho_1 \to \ldots \to \rho_p \to \iota$ we have decidable formulas TExt_ρ and Sep^i_ρ ($i = 1, \ldots, p$) such that

- $\forall_{U \in \mathrm{Con}_\rho}(U \subseteq \{\, a \mid \mathrm{TExt}_\rho(U, a) \,\} \in G_\rho)$ *and*
- $\forall_{U,V \in \mathrm{Con}_\rho}(U \not\uparrow_\rho V \to \vec{z}_{U,V} \in G \wedge U\vec{z}_{U,V} \not\uparrow_\iota V\vec{z}_{U,V})$,

where $\vec{z}_{U,V} = z_{U,V,1}, \ldots, z_{U,V,p}$ and $z_{U,V,i} = \{\, a \mid \mathrm{Sep}^i_\rho(U, V, a) \,\}$.

Proof.

By induction on ρ. $\square$

Definability

There will be two kinds of (natural) numbers:

- total tokens in **N**, i.e., $\mathrm{S}^n 0$ ("index numbers" $n \in \mathbb{N}$), and
- total ideals $\overline{n}$ of type **N**.

Fix enumerations

- $(e_n)_{n \in \mathbb{N}}$ of all tokens, and
- $(E_n)_{n \in \mathbb{N}}$ of all formal neighborhoods,

one for each type.

Definability

There will be two kinds of (natural) numbers:

- total tokens in $\mathbf{N}$, i.e., $\mathrm{S}^n 0$ ("index numbers" $n \in \mathbb{N}$), and
- total ideals $\overline{n}$ of type $\mathbf{N}$.

Fix enumerations

- $(e_n)_{n \in \mathbb{N}}$ of all tokens, and
- $(E_n)_{n \in \mathbb{N}}$ of all formal neighborhoods,

one for each type.

The parallel conditional $\mathrm{pcond}\colon \mathbf{B} \to \rho \to \rho \to \rho$

It is defined by the clauses

$$
\begin{aligned}
&U \vdash \mathtt{tt} \to V \vdash a \to (U, V, W, a) \in \mathrm{pcond},\\
&U \vdash \mathtt{ff} \to W \vdash a \to (U, V, W, a) \in \mathrm{pcond},\\
&V \vdash a \to W \vdash a \to (U, V, W, a) \in \mathrm{pcond}.
\end{aligned}
$$

We also need the least-fixed-point axiom, which says that any set of tokens (U, V, W, a) satisfying these is a superset of pcond.

Lemma (Properties of pcond)

pcond *is an ideal, and*

$$
\begin{aligned}
&\mathtt{tt} \in z \to \mathrm{pcond}(z, x, y) = x,\\
&\mathtt{ff} \in z \to \mathrm{pcond}(z, x, y) = y,\\
&a \in x \to a \in y \to a \in \mathrm{pcond}(z, x, y).
\end{aligned}
$$

The parallel conditional $\mathrm{pcond}\colon \mathbf{B} \to \rho \to \rho \to \rho$

It is defined by the clauses

$$U \vdash \mathtt{tt} \to V \vdash a \to (U, V, W, a) \in \mathrm{pcond},$$
$$U \vdash \mathtt{ff} \to W \vdash a \to (U, V, W, a) \in \mathrm{pcond},$$
$$V \vdash a \to W \vdash a \to (U, V, W, a) \in \mathrm{pcond}.$$

We also need the least-fixed-point axiom, which says that any set of tokens (U, V, W, a) satisfying these is a superset of pcond.

Lemma (Properties of pcond)

pcond *is an ideal, and*

$$\mathtt{tt} \in z \to \mathrm{pcond}(z, x, y) = x,$$
$$\mathtt{ff} \in z \to \mathrm{pcond}(z, x, y) = y,$$
$$a \in x \to a \in y \to a \in \mathrm{pcond}(z, x, y).$$

The parallel conditional $\mathrm{pcond}\colon \mathbf{B} \to \rho \to \rho \to \rho$

It is defined by the clauses

$$U \vdash \mathrm{tt} \to V \vdash a \to (U, V, W, a) \in \mathrm{pcond},$$
$$U \vdash \mathrm{ff} \to W \vdash a \to (U, V, W, a) \in \mathrm{pcond},$$
$$V \vdash a \to W \vdash a \to (U, V, W, a) \in \mathrm{pcond}.$$

We also need the least-fixed-point axiom, which says that any set of tokens (U, V, W, a) satisfying these is a superset of pcond.

Lemma (Properties of pcond)

pcond *is an ideal, and*

$$\mathrm{tt} \in z \to \mathrm{pcond}(z, x, y) = x,$$
$$\mathrm{ff} \in z \to \mathrm{pcond}(z, x, y) = y,$$
$$a \in x \to a \in y \to a \in \mathrm{pcond}(z, x, y).$$

A continuous variant of the union for **N**

For ideals in **N**, the union ($\sim$ maximum) is not continuous.
Continuous variant: $\cup^{\#}_{\mathbf{N}} \colon \mathbf{N} \to \mathbf{N} \to \mathbf{N}$, defined by the clauses

$$U \vdash e_n \to V \vdash n \to U \vdash a \to (U, V, a) \in \cup^{\#}_{\mathbf{N}},$$
$$\{e_n\} \vdash a \to V \vdash n \to (U, V, a) \in \cup^{\#}_{\mathbf{N}},$$

plus the least-fixed-point axiom.

Lemma (Properties of $\cup^{\#}_{\mathbf{N}}$)

$\cup^{\#}_{\mathbf{N}}$ *is an ideal, and*

$$\forall_{a \in x}(a \uparrow e_n) \to x \cup^{\#}_{\mathbf{N}} \overline{n} = x \cup \overline{\{e_n\}},$$
$$e_n \in x \cup^{\#}_{\mathbf{N}} \overline{n}.$$

A continuous variant of the union for $\mathbf{N}$

For ideals in $\mathbf{N}$, the union ($\sim$ maximum) is not continuous. Continuous variant: $\cup^{\#}_{\mathbf{N}} \colon \mathbf{N} \to \mathbf{N} \to \mathbf{N}$, defined by the clauses

$$U \vdash e_n \to V \vdash n \to U \vdash a \to (U, V, a) \in \cup^{\#}_{\mathbf{N}},$$
$$\{e_n\} \vdash a \to V \vdash n \to (U, V, a) \in \cup^{\#}_{\mathbf{N}},$$

plus the least-fixed-point axiom.

Lemma (Properties of $\cup^{\#}_{\mathbf{N}}$)

$\cup^{\#}_{\mathbf{N}}$ is an ideal, and

$$\forall_{a \in x}(a \uparrow e_n) \to x \cup^{\#}_{\mathbf{N}} \overline{n} = x \cup \overline{\{e_n\}},$$
$$e_n \in x \cup^{\#}_{\mathbf{N}} \overline{n}.$$

A continuous variant of consistency

Define $\uparrow^{\#}_{\rho}$ of type $\rho \to \mathbf{N} \to \mathbf{B}$ by the clauses

$$U \vdash E_n \to V \vdash n \to (U, V, \mathrm{tt}) \in \uparrow^{\#}_{\rho},$$
$$a \in U \to b \in E_n \to V \vdash n \to a \not\uparrow b \to (U, V, \mathrm{ff}) \in \uparrow^{\#}_{\rho} .$$

Again we require the least-fixed-point axiom.

Lemma (Properties of $\uparrow^{\#}_{\rho}$)

$\uparrow^{\#}_{\rho}$ *is an ideal, and*

$$\mathrm{tt} \in x \uparrow^{\#}_{\rho} \overline{n} \leftrightarrow x \supseteq E_n,$$
$$\mathrm{ff} \in x \uparrow^{\#}_{\rho} \overline{n} \leftrightarrow \exists_{a \in x, b \in E_n}(a \not\uparrow b).$$

A continuous variant of consistency

Define $\uparrow^{\#}_{\rho}$ of type $\rho \to \mathbf{N} \to \mathbf{B}$ by the clauses

$$U \vdash E_n \to V \vdash n \to (U, V, \mathtt{tt}) \in \uparrow^{\#}_{\rho},$$
$$a \in U \to b \in E_n \to V \vdash n \to a \not\uparrow b \to (U, V, \mathtt{ff}) \in \uparrow^{\#}_{\rho} .$$

Again we require the least-fixed-point axiom.

Lemma (Properties of $\uparrow^{\#}_{\rho}$)

$\uparrow^{\#}_{\rho}$ *is an ideal, and*

$$\mathtt{tt} \in x \uparrow^{\#}_{\rho} \overline{n} \leftrightarrow x \supseteq E_n,$$
$$\mathtt{ff} \in x \uparrow^{\#}_{\rho} \overline{n} \leftrightarrow \exists_{a \in x, b \in E_n}(a \not\uparrow b).$$

A continuous variant of existence

Define $\exists$ of type $(\mathbf{N} \to \mathbf{B}) \to \mathbf{B}$ by the clauses

$$U \vdash (\{\mathrm{S}^n 0\}, \mathrm{tt}) \to (U, \mathrm{tt}) \in \exists,$$
$$U \vdash (\{\mathrm{S}^n *\}, \mathrm{ff}) \to \forall_{i<n}(U \vdash (\{\mathrm{S}^i 0\}, \mathrm{ff})) \to (U, \mathrm{ff}) \in \exists,$$

plus the least-fixed-point axiom.

Lemma (Properties of $\exists$)

$\exists$ is an ideal, and

$$\mathrm{tt} \in \exists x \leftrightarrow \exists_n((\{\mathrm{S}^n 0\}, \mathrm{tt}) \in x),$$
$$\mathrm{ff} \in \exists x \leftrightarrow \exists_n((\{\mathrm{S}^n *\}, \mathrm{ff}) \in x \wedge \forall_{i<n}((\{\mathrm{S}^i 0\}, \mathrm{ff}) \in x).$$

A continuous variant of existence

Define $\exists$ of type $(\mathbf{N} \to \mathbf{B}) \to \mathbf{B}$ by the clauses

$$U \vdash (\{\mathrm{S}^n 0\}, \mathtt{tt}) \to (U, \mathtt{tt}) \in \exists,$$
$$U \vdash (\{\mathrm{S}^n *\}, \mathtt{ff}) \to \forall_{i<n}(U \vdash (\{\mathrm{S}^i 0\}, \mathtt{ff})) \to (U, \mathtt{ff}) \in \exists,$$

plus the least-fixed-point axiom.

Lemma (Properties of $\exists$)

$\exists$ *is an ideal, and*

$$\mathtt{tt} \in \exists x \leftrightarrow \exists_n((\{\mathrm{S}^n 0\}, \mathtt{tt}) \in x),$$
$$\mathtt{ff} \in \exists x \leftrightarrow \exists_n((\{\mathrm{S}^n *\}, \mathtt{ff}) \in x \land \forall_{i<n}((\{\mathrm{S}^i 0\}, \mathtt{ff}) \in x).$$

Definability

$\Phi\colon \rho \to \iota$ is called "recursive in $\cup^{\#}_{\mathbf{N}}$, pcond and $\uparrow^{\#}_{\rho}$" if it can be defined by a term involving the constructors for ι and $\mathbf{N}$, the fixed point operators Y_ρ, and predecessor, $\cup^{\#}_{\mathbf{N}}$, pcond and $\uparrow^{\#}_{\rho}$.

Theorem (Plotkin 1977)

For an algebra ι with at most unary constructors (e.g., $\mathbf{N}$, $\mathbf{B}$ or $\mathbf{P}$) and $\Phi\colon \rho \to \iota$ a partial continuous functional , the following are equivalent.

(a) Φ *is computable.*

(b) Φ *is recursive in* $\cup^{\#}_{\mathbf{N}}$, pcond *and* $\uparrow^{\#}_{\rho}$.

(c) Φ *is recursive in* $\cup^{\#}_{\mathbf{N}}$, pcond *and* $\exists$.

Definability

$\Phi \colon \rho \to \iota$ is called "recursive in $\cup^{\#}_{\mathbf{N}}$, pcond and $\uparrow^{\#}_{\rho}$" if it can be defined by a term involving the constructors for ι and $\mathbf{N}$, the fixed point operators Y_ρ, and predecessor, $\cup^{\#}_{\mathbf{N}}$, pcond and $\uparrow^{\#}_{\rho}$.

Theorem (Plotkin 1977)

For an algebra ι with at most unary constructors (e.g., $\mathbf{N}$, $\mathbf{B}$ or $\mathbf{P}$) and $\Phi \colon \rho \to \iota$ a partial continuous functional , the following are equivalent.

(a) Φ *is computable.*

(b) Φ *is recursive in* $\cup^{\#}_{\mathbf{N}}$, pcond *and* $\uparrow^{\#}_{\rho}$.

(c) Φ *is recursive in* $\cup^{\#}_{\mathbf{N}}$, pcond *and* $\exists$.

Proof of the definability theorem

$(a) \to (b)$. Let $\Phi\colon \rho \to \iota$ be computable:

$$\Phi = \{\, (E_{fn}, e_{gn}) \mid n \in \mathbb{N} \,\} \quad \text{with } f, g \text{ prim. rec. functions}$$

$\overline{f}$: continuous extension of f to ideals, such that $\overline{fn} = \overline{f}\,\overline{n}$. Show: Φ definable by $\Phi\varphi = Y w_\varphi \overline{0}$ with w_φ of type $(\mathbf{N} \to \iota) \to \mathbf{N} \to \iota$:

$$w_\varphi \psi x := \mathrm{pcond}(\varphi \uparrow^{\#}_{\rho} \overline{f}x, \psi(x+1) \cup^{\#}_{\mathbf{N}} \overline{g}x, \psi(x+1)).$$

Proof of the definability theorem (continued)

Write w for w_φ. Prove

$$\forall_n (a \in w^{k+1}\emptyset\overline{n} \to \exists_{n \le l \le n+k} (\varphi \supseteq E_{fl} \wedge \{e_{gl}\} \vdash a)). \qquad (1)$$

by induction on k. Step $k \mapsto k+1$:

$$a \in w^{k+2}\emptyset\overline{n} = w(w^{k+1}\emptyset)\overline{n} = \mathrm{pcond}(\varphi \uparrow^{\#}_{\rho} \overline{fn}, v \cup^{\#}_{\mathbf{N}} \overline{gn}, v),$$

with $v := w^{k+1}\emptyset(\overline{n}+1)$. Then either $a \in v$ ($\to$ done by IH) or else

$$\varphi \supseteq E_{fn} \wedge \{e_{gn}\} \vdash a.$$

Now $\Phi\varphi \supseteq Yw\overline{0}$ follows easily. Assume $a \in Yw\overline{0}$. Then $a \in w^{k+1}\emptyset\overline{0}$ for some k. By (1) there is an l with $0 \le l \le k$ such that $\varphi \supseteq E_{fl}$ and $\{e_{gl}\} \vdash a$. But this implies $a \in \Phi\varphi$.

Proof of the definability theorem (continued)

Converse: assume $a \in \Phi\varphi$. Then $(U, a) \in \Phi$ for some $U \subseteq \varphi$. By assumption on Φ: $U = E_{fn}$ and $a = e_{gn}$ for some n. We show

$$a \in w^{k+1}\emptyset(\overline{n-k}) \quad \text{for } k \leq n.$$

by induction on k. Step $k \mapsto k+1$: by definition of w $(:= w_\varphi)$

$$\begin{aligned} v' &:= w^{k+2}\emptyset(\overline{n-k-1}) \\ &= w(w^{k+1}\emptyset)(\overline{n-k-1}) \\ &= \mathrm{pcond}(\varphi \uparrow^{\#}_{\rho} \overline{f(n-k-1)}, v \cup^{\#}_{\mathbf{N}} \overline{g(n-k-1)}, v) \end{aligned}$$

with $v := w^{k+1}\emptyset(\overline{n-k})$. By IH: $a \in v$; we show $a \in v'$. If a and $e_{g(n-k-1)}$ are inconsistent, $a \in \Phi\varphi$ and $(E_{f(n-k-1)}, e_{g(n-k-1)}) \in \Phi$ imply that $\varphi \cup E_{f(n-k-1)}$ is inconsistent, hence $\mathrm{ff} \in \varphi \uparrow^{\#}_{\rho} \overline{f(n-k-1)}$ and therefore $v' = v$.

Proof of the definability theorem (continued)

If a and $e_{g(n-k-1)}$ are consistent, a and $e_{g(n-k-1)}$ are comparable, since the underlying algebra ι has at most unary constructors.

- $\{e_{g(n-k-1)}\} \vdash a$. Then $v \cup^{\#}_{\mathbf{N}} \overline{g(n-k-1)} \supseteq \{e_{g(n-k-1)}\} \vdash a$, and hence $a \in v'$ because of $a \in v$.
- $\{a\} \vdash e_{g(n-k-1)}$. Then $e_{g(n-k-1)} \in v$ because of $a \in v$, hence $v \cup^{\#}_{\mathbf{N}} \overline{g(n-k-1)} = v$ and therefore again $a \in v'$.

Now the converse inclusion $\Phi\varphi \subseteq Yw_{\varphi}\overline{0}$ can be seen easily. Since $a \in \Phi\varphi$, the claim just proved for $k := n$ gives $a \in w_{\varphi}^{n+1}\emptyset\overline{0}$, and this implies $a \in Yw_{\varphi}\overline{0}$.

TCF^+

- Theory of Computable Functionals plus their finite approximations, i.e., tokens and formal neighborhoods.
- Since continuous functionals (i.e., ideals) are possibly infinite sets of tokens, TCF^+ contains set variables x^ρ.
- The only existence axiom for sets will be Σ-comprehension.

TCF^+

- Theory of Computable Functionals plus their finite approximations, i.e., tokens and formal neighborhoods.
- Since continuous functionals (i.e., ideals) are possibly infinite sets of tokens, TCF^+ contains set variables x^ρ.
- The only existence axiom for sets will be Σ-comprehension.

TCF^+

- Theory of Computable Functionals plus their finite approximations, i.e., tokens and formal neighborhoods.
- Since continuous functionals (i.e., ideals) are possibly infinite sets of tokens, TCF^+ contains set variables x^ρ.
- The only existence axiom for sets will be Σ-comprehension.

Types and token types

Recall that (object) types are built from base types ι by $\rho \to \sigma$. In addition for every (object) type ρ we have token types (named τ):

- Tok^*_ρ (extended tokens of type ρ),
- Tok^*_ρ (tokens of type ρ),
- LTok_ρ (lists of tokens of type ρ),
- LTok^*_ρ (lists of extended tokens of type ρ).

We inductively define the extended tokens of $\mathbf{D}$, given by the constructors $0^{\mathbf{D}}$ (axiom) and $\mathrm{C}^{\mathbf{D}\to\mathbf{D}\to\mathbf{D}}$ (rule). The clauses are

$$\mathrm{Tok}^*_{\mathbf{D}}(*), \quad \mathrm{Tok}^*_{\mathbf{D}}(0^{\mathbf{D}}),$$
$$\mathrm{Tok}^*_{\mathbf{D}}(a_1^*) \to \mathrm{Tok}^*_{\mathbf{D}}(a_2^*) \to \mathrm{Tok}^*_{\mathbf{D}}(\mathrm{C}^{\mathbf{D}\to\mathbf{D}\to\mathbf{D}} a_1^* a_2^*).$$

Similarly for Tok^*_ρ, LTok_ρ, LTok^*_ρ.

Types and token types

Recall that (object) types are built from base types ι by $\rho \to \sigma$. In addition for every (object) type ρ we have token types (named τ):

- Tok^*_ρ (extended tokens of type ρ),
- Tok^*_ρ (tokens of type ρ),
- LTok_ρ (lists of tokens of type ρ),
- LTok^*_ρ (lists of extended tokens of type ρ).

We inductively define the extended tokens of $\mathbf{D}$, given by the constructors $0^{\mathbf{D}}$ (axiom) and $\mathrm{C}^{\mathbf{D}\to\mathbf{D}\to\mathbf{D}}$ (rule). The clauses are

$$\mathrm{Tok}^*_{\mathbf{D}}(*), \quad \mathrm{Tok}^*_{\mathbf{D}}(0^{\mathbf{D}}),$$

$$\mathrm{Tok}^*_{\mathbf{D}}(a^*_1) \to \mathrm{Tok}^*_{\mathbf{D}}(a^*_2) \to \mathrm{Tok}^*_{\mathbf{D}}(\mathrm{C}^{\mathbf{D}\to\mathbf{D}\to\mathbf{D}} a^*_1 a^*_2).$$

Similarly for Tok^*_ρ, LTok_ρ, LTok^*_ρ.

Types and token types

Recall that (object) types are built from base types ι by $\rho \to \sigma$. In addition for every (object) type ρ we have token types (named τ):

- Tok^*_ρ (extended tokens of type ρ),
- Tok^*_ρ (tokens of type ρ),
- LTok_ρ (lists of tokens of type ρ),
- LTok^*_ρ (lists of extended tokens of type ρ).

We inductively define the extended tokens of **D**, given by the constructors $0^{\mathbf{D}}$ (axiom) and $\mathrm{C}^{\mathbf{D}\to\mathbf{D}\to\mathbf{D}}$ (rule). The clauses are

$$\mathrm{Tok}^*_{\mathbf{D}}(*), \quad \mathrm{Tok}^*_{\mathbf{D}}(0^{\mathbf{D}}),$$
$$\mathrm{Tok}^*_{\mathbf{D}}(a^*_1) \to \mathrm{Tok}^*_{\mathbf{D}}(a^*_2) \to \mathrm{Tok}^*_{\mathbf{D}}(\mathrm{C}^{\mathbf{D}\to\mathbf{D}\to\mathbf{D}} a^*_1 a^*_2).$$

Similarly for Tok^*_ρ, LTok_ρ, LTok^*_ρ.

Types and token types

Recall that (object) types are built from base types ι by $\rho \to \sigma$. In addition for every (object) type ρ we have token types (named τ):

- Tok^*_ρ (extended tokens of type ρ),
- Tok^*_ρ (tokens of type ρ),
- LTok_ρ (lists of tokens of type ρ),
- LTok^*_ρ (lists of extended tokens of type ρ).

We inductively define the extended tokens of $\mathbf{D}$, given by the constructors $0^{\mathbf{D}}$ (axiom) and $\mathrm{C}^{\mathbf{D}\to\mathbf{D}\to\mathbf{D}}$ (rule). The clauses are

$$\mathrm{Tok}^*_{\mathbf{D}}(*), \quad \mathrm{Tok}^*_{\mathbf{D}}(0^{\mathbf{D}}),$$
$$\mathrm{Tok}^*_{\mathbf{D}}(a^*_1) \to \mathrm{Tok}^*_{\mathbf{D}}(a^*_2) \to \mathrm{Tok}^*_{\mathbf{D}}(\mathrm{C}^{\mathbf{D}\to\mathbf{D}\to\mathbf{D}} a^*_1 a^*_2).$$

Similarly for Tok^*_ρ, LTok_ρ, LTok^*_ρ.

Functions of token-valued types $\vec{\tau} \to \tau$

Example: $\dot{\in}_{\mathbf{D}} \colon \mathrm{Tok}^*_{\mathbf{D}} \to \mathrm{LTok}^*_{\mathbf{D}} \to \mathrm{Tok}_{\mathbf{B}}$. Recursion equations:

$$(a^* \mathrel{\dot{\in}_{\mathbf{D}}} \mathrm{nil}) := \mathrm{ff},$$
$$(a^* \mathrel{\dot{\in}_{\mathbf{D}}} (b^* ::_{\mathbf{D}} U)) := (a^* =_{\mathbf{D}} b^*) \vee_{\mathbf{B}} a^* \mathrel{\dot{\in}} U,$$

where equality $=_{\mathbf{D}} \colon \mathrm{Tok}^*_{\mathbf{D}} \to \mathrm{Tok}^*_{\mathbf{D}} \to \mathrm{Tok}_{\mathbf{B}}$ is defined by

$$(* =_{\mathbf{D}} *) := (0 =_{\mathbf{D}} 0) := \mathrm{tt},$$
$$(* =_{\mathbf{D}} 0) := (* =_{\mathbf{D}} \mathrm{C}a_1^* a_2^*) := \mathrm{ff},$$
$$(0 =_{\mathbf{D}} *) := (0 =_{\mathbf{D}} \mathrm{C}a_1^* a_2^*) := \mathrm{ff},$$
$$(\mathrm{C}a_1^* a_2^* =_{\mathbf{D}} *) := (\mathrm{C}a_1^* a_2^* =_{\mathbf{D}} 0) := \mathrm{ff},$$
$$(\mathrm{C}a_1^* a_2^* =_{\mathbf{D}} \mathrm{C}b_1^* b_2^*) := (a_1^* =_{\mathbf{D}} b_1^*) \wedge_{\mathbf{B}} (a_2^* =_{\mathbf{D}} b_2^*),$$

and $\vee_{\mathbf{B}}, \wedge_{\mathbf{B}} \colon \mathrm{Tok}_{\mathbf{B}} \to \mathrm{Tok}_{\mathbf{B}} \to \mathrm{Tok}_{\mathbf{B}}$ are defined by $\mathrm{tt} \vee_{\mathbf{B}} b := \mathrm{tt}$, $\mathrm{ff} \vee_{\mathbf{B}} b := b$, $\mathrm{ff} \wedge_{\mathbf{B}} b := \mathrm{ff}$ and $\mathrm{tt} \wedge_{\mathbf{B}} b := b$.
Similarly: $\vdash \colon \mathrm{LTok}_{\mathbf{D}} \to \mathrm{Tok}^*_{\mathbf{D}} \to \mathrm{Tok}_{\mathbf{B}}$, $\mathrm{Con} \colon \mathrm{LTok}_{\mathbf{D}} \to \mathrm{Tok}_{\mathbf{B}}$ etc.

Functions of token-valued types $\vec{\tau} \to \tau$

Example: $\dot{\in}_{\mathbf{D}} \colon \mathrm{Tok}^*_{\mathbf{D}} \to \mathrm{LTok}^*_{\mathbf{D}} \to \mathrm{Tok}_{\mathbf{B}}$. Recursion equations:

$$\begin{aligned}
(a^* \mathrel{\dot{\in}_{\mathbf{D}}} \mathrm{nil}) &:= \mathrm{ff}, \\
(a^* \mathrel{\dot{\in}_{\mathbf{D}}} (b^* ::_{\mathbf{D}} U)) &:= (a^* =_{\mathbf{D}} b^*) \vee_{\mathbf{B}} a^* \mathrel{\dot{\in}} U,
\end{aligned}$$

where equality $=_{\mathbf{D}} \colon \mathrm{Tok}^*_{\mathbf{D}} \to \mathrm{Tok}^*_{\mathbf{D}} \to \mathrm{Tok}_{\mathbf{B}}$ is defined by

$$\begin{aligned}
&(* =_{\mathbf{D}} *) := (0 =_{\mathbf{D}} 0) := \mathrm{tt}, \\
&(* =_{\mathbf{D}} 0) := (* =_{\mathbf{D}} \mathrm{C} a_1^* a_2^*) := \mathrm{ff}, \\
&(0 =_{\mathbf{D}} *) := (0 =_{\mathbf{D}} \mathrm{C} a_1^* a_2^*) := \mathrm{ff}, \\
&(\mathrm{C} a_1^* a_2^* =_{\mathbf{D}} *) := (\mathrm{C} a_1^* a_2^* =_{\mathbf{D}} 0) := \mathrm{ff}, \\
&(\mathrm{C} a_1^* a_2^* =_{\mathbf{D}} \mathrm{C} b_1^* b_2^*) := (a_1^* =_{\mathbf{D}} b_1^*) \wedge_{\mathbf{B}} (a_2^* =_{\mathbf{D}} b_2^*),
\end{aligned}$$

and $\vee_{\mathbf{B}}, \wedge_{\mathbf{B}} \colon \mathrm{Tok}_{\mathbf{B}} \to \mathrm{Tok}_{\mathbf{B}} \to \mathrm{Tok}_{\mathbf{B}}$ are defined by $\mathrm{tt} \vee_{\mathbf{B}} b := \mathrm{tt}$, $\mathrm{ff} \vee_{\mathbf{B}} b := b$, $\mathrm{ff} \wedge_{\mathbf{B}} b := \mathrm{ff}$ and $\mathrm{tt} \wedge_{\mathbf{B}} b := b$.

Similarly: $\vdash \colon \mathrm{LTok}_{\mathbf{D}} \to \mathrm{Tok}^*_{\mathbf{D}} \to \mathrm{Tok}_{\mathbf{B}}$, $\mathrm{Con} \colon \mathrm{LTok}_{\mathbf{D}} \to \mathrm{Tok}_{\mathbf{B}}$ etc.

Functions of token-valued types $\vec{\tau} \to \tau$

Example: $\dot{\in}_{\mathbf{D}} \colon \mathrm{Tok}^*_{\mathbf{D}} \to \mathrm{LTok}^*_{\mathbf{D}} \to \mathrm{Tok}_{\mathbf{B}}$. Recursion equations:

$$
\begin{aligned}
(a^* \mathrel{\dot{\in}_{\mathbf{D}}} \mathrm{nil}) &:= \mathrm{ff}, \\
(a^* \mathrel{\dot{\in}_{\mathbf{D}}} (b^* ::_{\mathbf{D}} U)) &:= (a^* =_{\mathbf{D}} b^*) \vee_{\mathbf{B}} a^* \mathrel{\dot{\in}} U,
\end{aligned}
$$

where equality $=_{\mathbf{D}} \colon \mathrm{Tok}^*_{\mathbf{D}} \to \mathrm{Tok}^*_{\mathbf{D}} \to \mathrm{Tok}_{\mathbf{B}}$ is defined by

$$
\begin{aligned}
(* =_{\mathbf{D}} *) &:= (0 =_{\mathbf{D}} 0) := \mathrm{tt}, \\
(* =_{\mathbf{D}} 0) &:= (* =_{\mathbf{D}} \mathrm{C} a_1^* a_2^*) := \mathrm{ff}, \\
(0 =_{\mathbf{D}} *) &:= (0 =_{\mathbf{D}} \mathrm{C} a_1^* a_2^*) := \mathrm{ff}, \\
(\mathrm{C} a_1^* a_2^* =_{\mathbf{D}} *) &:= (\mathrm{C} a_1^* a_2^* =_{\mathbf{D}} 0) := \mathrm{ff}, \\
(\mathrm{C} a_1^* a_2^* =_{\mathbf{D}} \mathrm{C} b_1^* b_2^*) &:= (a_1^* =_{\mathbf{D}} b_1^*) \wedge_{\mathbf{B}} (a_2^* =_{\mathbf{D}} b_2^*),
\end{aligned}
$$

and $\vee_{\mathbf{B}}, \wedge_{\mathbf{B}} \colon \mathrm{Tok}_{\mathbf{B}} \to \mathrm{Tok}_{\mathbf{B}} \to \mathrm{Tok}_{\mathbf{B}}$ are defined by $\mathrm{tt} \vee_{\mathbf{B}} b := \mathrm{tt}$, $\mathrm{ff} \vee_{\mathbf{B}} b := b$, $\mathrm{ff} \wedge_{\mathbf{B}} b := \mathrm{ff}$ and $\mathrm{tt} \wedge_{\mathbf{B}} b := b$.
Similarly: $\vdash \colon \mathrm{LTok}_{\mathbf{D}} \to \mathrm{Tok}^*_{\mathbf{D}} \to \mathrm{Tok}_{\mathbf{B}}$, $\mathrm{Con} \colon \mathrm{LTok}_{\mathbf{D}} \to \mathrm{Tok}_{\mathbf{B}}$ etc.

Tokens of higher type

Tokens of a function type $\rho \to \sigma$ are pairs (U, a) of lists of tokens of type ρ and tokens of type σ. Both projections are given by functions π_1, π_2. Consistency of lists of tokens, application WU and entailment $W \vdash (U, a)$ can be defined as described as above.

- Variables a^* for Tok^*_ρ, a for Tok_ρ, U for LTok_ρ.
- From these, the symbols for token-valued functions and constants for the constructors for tokens, extended tokens and lists of these we can build terms of token types.
- We identify terms of token type if they have the same normal form w.r.t. the defining primitive recursion equations for the token-valued functions involved.

Tokens of higher type

Tokens of a function type $\rho \to \sigma$ are pairs (U, a) of lists of tokens of type ρ and tokens of type σ. Both projections are given by functions π_1, π_2. Consistency of lists of tokens, application WU and entailment $W \vdash (U, a)$ can be defined as described as above.

- Variables a^* for Tok^*_ρ, a for Tok_ρ, U for LTok_ρ.
- From these, the symbols for token-valued functions and constants for the constructors for tokens, extended tokens and lists of these we can build terms of token types.
- We identify terms of token type if they have the same normal form w.r.t. the defining primitive recursion equations for the token-valued functions involved.

Tokens of higher type

Tokens of a function type $\rho \to \sigma$ are pairs (U, a) of lists of tokens of type ρ and tokens of type σ. Both projections are given by functions π_1, π_2. Consistency of lists of tokens, application WU and entailment $W \vdash (U, a)$ can be defined as described as above.

- Variables a^* for Tok^*_ρ, a for Tok_ρ, U for LTok_ρ.
- From these, the symbols for token-valued functions and constants for the constructors for tokens, extended tokens and lists of these we can build terms of token types.
- We identify terms of token type if they have the same normal form w.r.t. the defining primitive recursion equations for the token-valued functions involved.

Tokens of higher type

Tokens of a function type $\rho \to \sigma$ are pairs (U, a) of lists of tokens of type ρ and tokens of type σ. Both projections are given by functions π_1, π_2. Consistency of lists of tokens, application WU and entailment $W \vdash (U, a)$ can be defined as described as above.

- Variables a^* for Tok^*_ρ, a for Tok_ρ, U for LTok_ρ.
- From these, the symbols for token-valued functions and constants for the constructors for tokens, extended tokens and lists of these we can build terms of token types.
- We identify terms of token type if they have the same normal form w.r.t. the defining primitive recursion equations for the token-valued functions involved.

Tokens of higher type

Tokens of a function type $\rho \to \sigma$ are pairs (U, a) of lists of tokens of type ρ and tokens of type σ. Both projections are given by functions π_1, π_2. Consistency of lists of tokens, application WU and entailment $W \vdash (U, a)$ can be defined as described as above.

- Variables a^* for Tok^*_ρ, a for Tok_ρ, U for LTok_ρ.
- From these, the symbols for token-valued functions and constants for the constructors for tokens, extended tokens and lists of these we can build terms of token types.
- We identify terms of token type if they have the same normal form w.r.t. the defining primitive recursion equations for the token-valued functions involved.

Tokens of higher type

Tokens of a function type $\rho \to \sigma$ are pairs (U, a) of lists of tokens of type ρ and tokens of type σ. Both projections are given by functions π_1, π_2. Consistency of lists of tokens, application WU and entailment $W \vdash (U, a)$ can be defined as described as above.

- Variables a^* for Tok^*_ρ, a for Tok_ρ, U for LTok_ρ.
- From these, the symbols for token-valued functions and constants for the constructors for tokens, extended tokens and lists of these we can build terms of token types.
- We identify terms of token type if they have the same normal form w.r.t. the defining primitive recursion equations for the token-valued functions involved.

Formulas

- Prime Δ-formulas: $\mathrm{atom}(p)$, with p term of token type $\mathrm{Tok}_{\mathbf{B}}$. Examples: $a \uparrow_\rho b$, $a \mathrel{\dot\in}_\rho U$, $U \vdash_\rho a$ (i.e., $\mathrm{atom}(a \uparrow_\rho b)$ etc.)
- Δ-formulas: from prime Δ-formulas by $\to, \wedge, \vee, \forall_{a \dot\in U}, \exists_{a \dot\in U}$, with a a variable for tokens and U a term for a list of tokens.
- Variables x^ρ and constants of (object) type ρ, intended to denote sets of tokens. Constants: $[\![\lambda_{\vec{x}} M]\!]$, $\cup^{\#}_{\mathbf{N}}$, pcond, $\uparrow^{\#}_{\rho}$.
- Prime Σ-formulas: prime Δ-formulas or of the form $r \in_\rho x$, with $r \colon \mathrm{Tok}_\rho$ a term and x a variable or constant of type ρ.
- Σ-formulas: (i) prime Σ-formulas, (ii) $A_0 \to B$ with A_0 a Δ- and B a Σ-formula, and (iii) closed under $\wedge, \vee, \forall_{a \dot\in U}, \exists_{a \dot\in U}$ and existential quantifiers over variables of a token type.
- Prime formulas: prime Σ-formulas or $G_\rho x$ (totality of x) or $x \approx_\rho y$ (equivalence of x and y); x, y variables or constants.
- Formulas: from prime formulas by $\to, \wedge, \vee, \forall, \exists$.

Formulas

- Prime Δ-formulas: $\mathrm{atom}(p)$, with p term of token type $\mathrm{Tok}_{\mathbf{B}}$. Examples: $a \uparrow_\rho b$, $a \,\dot{\in}_\rho\, U$, $U \vdash_\rho a$ (i.e., $\mathrm{atom}(a \uparrow_\rho b)$ etc.)
- Δ-formulas: from prime Δ-formulas by $\to, \wedge, \vee, \forall_{a \dot{\in} U}, \exists_{a \dot{\in} U}$, with a a variable for tokens and U a term for a list of tokens.
- Variables x^ρ and constants of (object) type ρ, intended to denote sets of tokens. Constants: $[\![\lambda_{\vec{x}} M]\!]$, $\cup^{\#}_{\mathbf{N}}$, pcond, $\uparrow^{\#}_\rho$.
- Prime Σ-formulas: prime Δ-formulas or of the form $r \in_\rho x$, with $r \colon \mathrm{Tok}_\rho$ a term and x a variable or constant of type ρ.
- Σ-formulas: (i) prime Σ-formulas, (ii) $A_0 \to B$ with A_0 a Δ- and B a Σ-formula, and (iii) closed under $\wedge, \vee, \forall_{a \dot{\in} U}, \exists_{a \dot{\in} U}$ and existential quantifiers over variables of a token type.
- Prime formulas: prime Σ-formulas or $G_\rho x$ (totality of x) or $x \approx_\rho y$ (equivalence of x and y); x, y variables or constants.
- Formulas: from prime formulas by $\to, \wedge, \vee, \forall, \exists$.

Formulas

- Prime Δ-formulas: $\mathrm{atom}(p)$, with p term of token type $\mathrm{Tok}_{\mathbf{B}}$. Examples: $a \uparrow_\rho b$, $a \,\dot{\in}_\rho\, U$, $U \vdash_\rho a$ (i.e., $\mathrm{atom}(a \uparrow_\rho b)$ etc.)
- Δ-formulas: from prime Δ-formulas by $\to, \land, \lor, \forall_{a \dot{\in} U}, \exists_{a \dot{\in} U}$, with a a variable for tokens and U a term for a list of tokens.
- Variables x^ρ and constants of (object) type ρ, intended to denote sets of tokens. Constants: $[\![\lambda_{\vec{x}} M]\!]$, $\cup^{\#}_{\mathbf{N}}$, pcond, $\uparrow^{\#}_\rho$.
- Prime Σ-formulas: prime Δ-formulas or of the form $r \in_\rho x$, with $r \colon \mathrm{Tok}_\rho$ a term and x a variable or constant of type ρ.
- Σ-formulas: (i) prime Σ-formulas, (ii) $A_0 \to B$ with A_0 a Δ- and B a Σ-formula, and (iii) closed under $\land, \lor, \forall_{a \dot{\in} U}, \exists_{a \dot{\in} U}$ and existential quantifiers over variables of a token type.
- Prime formulas: prime Σ-formulas or $G_\rho x$ (totality of x) or $x \approx_\rho y$ (equivalence of x and y); x, y variables or constants.
- Formulas: from prime formulas by $\to, \land, \lor, \forall, \exists$.

Formulas

- Prime Δ-formulas: $\mathrm{atom}(p)$, with p term of token type $\mathrm{Tok}_{\mathbf{B}}$. Examples: $a \uparrow_\rho b$, $a \dot{\in}_\rho U$, $U \vdash_\rho a$ (i.e., $\mathrm{atom}(a \uparrow_\rho b)$ etc.)
- Δ-formulas: from prime Δ-formulas by $\to, \land, \lor, \forall_{a \dot{\in} U}, \exists_{a \dot{\in} U}$, with a a variable for tokens and U a term for a list of tokens.
- Variables x^ρ and constants of (object) type ρ, intended to denote sets of tokens. Constants: $[\![\lambda_{\vec{x}} M]\!]$, $\cup^{\#}_{\mathbf{N}}$, pcond, $\uparrow^{\#}_{\rho}$.
- Prime Σ-formulas: prime Δ-formulas or of the form $r \in_\rho x$, with $r \colon \mathrm{Tok}_\rho$ a term and x a variable or constant of type ρ.
- Σ-formulas: (i) prime Σ-formulas, (ii) $A_0 \to B$ with A_0 a Δ- and B a Σ-formula, and (iii) closed under $\land, \lor, \forall_{a \dot{\in} U}, \exists_{a \dot{\in} U}$ and existential quantifiers over variables of a token type.
- Prime formulas: prime Σ-formulas or $G_\rho x$ (totality of x) or $x \approx_\rho y$ (equivalence of x and y); x, y variables or constants.
- Formulas: from prime formulas by $\to, \land, \lor, \forall, \exists$.

Formulas

- Prime Δ-formulas: $\mathrm{atom}(p)$, with p term of token type $\mathrm{Tok}_{\mathbf{B}}$. Examples: $a \uparrow_\rho b$, $a \mathrel{\dot\in}_\rho U$, $U \vdash_\rho a$ (i.e., $\mathrm{atom}(a \uparrow_\rho b)$ etc.)
- Δ-formulas: from prime Δ-formulas by $\to, \wedge, \vee, \forall_{a \dot\in U}, \exists_{a \dot\in U}$, with a a variable for tokens and U a term for a list of tokens.
- Variables x^ρ and constants of (object) type ρ, intended to denote sets of tokens. Constants: $[\![\lambda_{\vec{x}} M]\!]$, $\cup^{\#}_{\mathbf{N}}$, pcond, $\uparrow^{\#}_\rho$.
- Prime Σ-formulas: prime Δ-formulas or of the form $r \in_\rho x$, with $r\colon \mathrm{Tok}_\rho$ a term and x a variable or constant of type ρ.
- Σ-formulas: (i) prime Σ-formulas, (ii) $A_0 \to B$ with A_0 a Δ- and B a Σ-formula, and (iii) closed under $\wedge, \vee, \forall_{a \dot\in U}, \exists_{a \dot\in U}$ and existential quantifiers over variables of a token type.
- Prime formulas: prime Σ-formulas or $G_\rho x$ (totality of x) or $x \approx_\rho y$ (equivalence of x and y); x, y variables or constants.
- Formulas: from prime formulas by $\to, \wedge, \vee, \forall, \exists$.

Formulas

- Prime Δ-formulas: $\mathrm{atom}(p)$, with p term of token type $\mathrm{Tok}_{\mathbf{B}}$. Examples: $a \uparrow_\rho b$, $a \mathrel{\dot\in}_\rho U$, $U \vdash_\rho a$ (i.e., $\mathrm{atom}(a \uparrow_\rho b)$ etc.)
- Δ-formulas: from prime Δ-formulas by $\to, \wedge, \vee, \forall_{a \dot\in U}, \exists_{a \dot\in U}$, with a a variable for tokens and U a term for a list of tokens.
- Variables x^ρ and constants of (object) type ρ, intended to denote sets of tokens. Constants: $[\![\lambda_{\vec{x}} M]\!]$, $\cup^{\#}_{\mathbf{N}}$, pcond, $\uparrow^{\#}_\rho$.
- Prime Σ-formulas: prime Δ-formulas or of the form $r \in_\rho x$, with $r \colon \mathrm{Tok}_\rho$ a term and x a variable or constant of type ρ.
- Σ-formulas: (i) prime Σ-formulas, (ii) $A_0 \to B$ with A_0 a Δ- and B a Σ-formula, and (iii) closed under $\wedge, \vee, \forall_{a \dot\in U}, \exists_{a \dot\in U}$ and existential quantifiers over variables of a token type.
- Prime formulas: prime Σ-formulas or $G_\rho x$ (totality of x) or $x \approx_\rho y$ (equivalence of x and y); x, y variables or constants.
- Formulas: from prime formulas by $\to, \wedge, \vee, \forall, \exists$.

Formulas

- Prime Δ-formulas: $\mathrm{atom}(p)$, with p term of token type $\mathrm{Tok}_{\mathbf{B}}$. Examples: $a \uparrow_\rho b$, $a \mathrel{\dot\in}_\rho U$, $U \vdash_\rho a$ (i.e., $\mathrm{atom}(a \uparrow_\rho b)$ etc.)
- Δ-formulas: from prime Δ-formulas by $\to, \land, \lor, \forall_{a \dot\in U}, \exists_{a \dot\in U}$, with a a variable for tokens and U a term for a list of tokens.
- Variables x^ρ and constants of (object) type ρ, intended to denote sets of tokens. Constants: $[\![\lambda_{\vec{x}} M]\!]$, $\cup^{\#}_{\mathbf{N}}$, pcond, $\uparrow^{\#}_\rho$.
- Prime Σ-formulas: prime Δ-formulas or of the form $r \in_\rho x$, with $r \colon \mathrm{Tok}_\rho$ a term and x a variable or constant of type ρ.
- Σ-formulas: (i) prime Σ-formulas, (ii) $A_0 \to B$ with A_0 a Δ- and B a Σ-formula, and (iii) closed under $\land, \lor, \forall_{a \dot\in U}, \exists_{a \dot\in U}$ and existential quantifiers over variables of a token type.
- Prime formulas: prime Σ-formulas or $G_\rho x$ (totality of x) or $x \approx_\rho y$ (equivalence of x and y); x, y variables or constants.
- Formulas: from prime formulas by $\to, \land, \lor, \forall, \exists$.

Axioms of TCF^+

- Based on minimal logic. Define $\mathbf{F} := \mathrm{atom}(\mathrm{ff})$ ("falsum").
- $\mathbf{F} \to A$ ("ex-falso-quodlibet") for prime non-Δ prime formulas.
- Usual axioms of Heyting arithmetic, adapted to token types:

$$A(\mathrm{tt}) \to A(\mathrm{ff}) \to A(a),$$
$$A(*) \to A(0) \to \forall_{a^*,b^*}(A(a^*) \to A(b^*) \to A(\mathrm{C}a^*b^*)) \to A(a^*).$$

- $\mathrm{atom}(\mathrm{tt})$.
- $\exists_x \forall_a (a \in_\rho x \leftrightarrow A)$ for A Σ-formula (ρ an object type)
- For every constant $[\![\lambda_{\vec{x}} M]\!]$, (V), (A), (C), (D), + lfp axioms.
- Defining clauses and lfp axioms for $\cup^{\#}_{\mathbf{N}}$, pcond, $\uparrow^{\#}_{\rho}$, $\exists$.
- The clauses defining the totality predicates G_ρ and the equivalence relations $x_1 \approx_\rho x_2$, together with their lfp axioms.

Theorem

TCF^+ *proves the density theorem and the definability theorem.*

Axioms of TCF^+

- Based on minimal logic. Define $\mathbf{F} := \mathrm{atom}(\mathrm{ff})$ ("falsum").
- $\mathbf{F} \to A$ ("ex-falso-quodlibet") for prime non-Δ prime formulas.
- Usual axioms of Heyting arithmetic, adapted to token types:

$$A(\mathrm{tt}) \to A(\mathrm{ff}) \to A(a),$$
$$A(*) \to A(0) \to \forall_{a^*,b^*}(A(a^*) \to A(b^*) \to A(\mathrm{C}a^*b^*)) \to A(a^*).$$

- $\mathrm{atom}(\mathrm{tt})$.
- $\exists_x \forall_a (a \in_\rho x \leftrightarrow A)$ for A Σ-formula (ρ an object type)
- For every constant $[\![\lambda_{\vec{x}} M]\!]$, (V), (A), (C), (D), + lfp axioms.
- Defining clauses and lfp axioms for $\cup^{\#}_{\mathbf{N}}$, pcond, $\uparrow^{\#}_{\rho}$, $\exists$.
- The clauses defining the totality predicates G_ρ and the equivalence relations $x_1 \approx_\rho x_2$, together with their lfp axioms.

Theorem

TCF^+ *proves the density theorem and the definability theorem.*

Axioms of TCF^+

- Based on minimal logic. Define $\mathbf{F} := \mathrm{atom}(\mathrm{ff})$ ("falsum").
- $\mathbf{F} \to A$ ("ex-falso-quodlibet") for prime non-Δ prime formulas.
- Usual axioms of Heyting arithmetic, adapted to token types:

$$A(\mathrm{tt}) \to A(\mathrm{ff}) \to A(a),$$
$$A(*) \to A(0) \to \forall_{a^*,b^*}(A(a^*) \to A(b^*) \to A(\mathrm{C}a^*b^*)) \to A(a^*).$$

- $\mathrm{atom}(\mathrm{tt})$.
- $\exists_x \forall_a (a \in_\rho x \leftrightarrow A)$ for A Σ-formula (ρ an object type)
- For every constant $[\![\lambda_{\vec{x}} M]\!]$, (V), (A), (C), (D), + lfp axioms.
- Defining clauses and lfp axioms for $\cup^{\#}_{\mathbf{N}}$, pcond, $\uparrow^{\#}_{\rho}$, $\exists$.
- The clauses defining the totality predicates G_ρ and the equivalence relations $x_1 \approx_\rho x_2$, together with their lfp axioms.

Theorem

TCF^+ *proves the density theorem and the definability theorem.*

Axioms of TCF^+

- Based on minimal logic. Define $\mathbf{F} := \mathrm{atom}(\mathit{ff})$ ("falsum").
- $\mathbf{F} \to A$ ("ex-falso-quodlibet") for prime non-Δ prime formulas.
- Usual axioms of Heyting arithmetic, adapted to token types:

$$A(\mathit{tt}) \to A(\mathit{ff}) \to A(a),$$
$$A(*) \to A(0) \to \forall_{a^*,b^*}(A(a^*) \to A(b^*) \to A(\mathrm{C}a^*b^*)) \to A(a^*).$$

- $\mathrm{atom}(\mathit{tt})$.
- $\exists_x \forall_a (a \in_\rho x \leftrightarrow A)$ for A Σ-formula (ρ an object type)
- For every constant $[\![\lambda_{\vec{x}} M]\!]$, (V), (A), (C), (D), + lfp axioms.
- Defining clauses and lfp axioms for $\cup^{\#}_{\mathbf{N}}$, pcond, $\uparrow^{\#}_{\rho}$, $\exists$.
- The clauses defining the totality predicates G_ρ and the equivalence relations $x_1 \approx_\rho x_2$, together with their lfp axioms.

Theorem

TCF^+ *proves the density theorem and the definability theorem.*

Axioms of TCF^{+}

- Based on minimal logic. Define $\mathbf{F} := \mathrm{atom}(\mathit{ff})$ ("falsum").
- $\mathbf{F} \to A$ ("ex-falso-quodlibet") for prime non-Δ prime formulas.
- Usual axioms of Heyting arithmetic, adapted to token types:

$$A(\mathit{tt}) \to A(\mathit{ff}) \to A(a),$$
$$A(*) \to A(0) \to \forall_{a^*,b^*}(A(a^*) \to A(b^*) \to A(\mathrm{C}a^* b^*)) \to A(a^*).$$

- $\mathrm{atom}(\mathit{tt})$.
- $\exists_x \forall_a (a \in_\rho x \leftrightarrow A)$ for A Σ-formula (ρ an object type)
- For every constant $[\![\lambda_{\vec{x}} M]\!]$, (V), (A), (C), (D), $+$ lfp axioms.
- Defining clauses and lfp axioms for $\cup^{\#}_{\mathbf{N}}$, pcond, $\uparrow^{\#}_{\rho}$, $\exists$.
- The clauses defining the totality predicates G_ρ and the equivalence relations $x_1 \approx_\rho x_2$, together with their lfp axioms.

Theorem

TCF^{+} *proves the density theorem and the definability theorem.*

Axioms of TCF^+

- Based on minimal logic. Define $\mathbf{F} := \mathrm{atom}(\mathit{ff})$ ("falsum").
- $\mathbf{F} \to A$ ("ex-falso-quodlibet") for prime non-Δ prime formulas.
- Usual axioms of Heyting arithmetic, adapted to token types:

$$A(\mathit{tt}) \to A(\mathit{ff}) \to A(a),$$
$$A(*) \to A(0) \to \forall_{a^*,b^*}(A(a^*) \to A(b^*) \to A(\mathrm{C}a^* b^*)) \to A(a^*).$$

- $\mathrm{atom}(\mathit{tt})$.
- $\exists_x \forall_a (a \in_\rho x \leftrightarrow A)$ for A Σ-formula (ρ an object type)
- For every constant $[\![\lambda_{\vec{x}} M]\!]$, (V), (A), (C), (D), + lfp axioms.
- Defining clauses and lfp axioms for $\cup^{\#}_{\mathbf{N}}$, pcond, $\uparrow^{\#}_{\rho}$, $\exists$.
- The clauses defining the totality predicates G_ρ and the equivalence relations $x_1 \approx_\rho x_2$, together with their lfp axioms.

Theorem

TCF^+ *proves the density theorem and the definability theorem.*

Axioms of TCF^+

- Based on minimal logic. Define $\mathbf{F} := \mathrm{atom}(\mathit{ff})$ ("falsum").
- $\mathbf{F} \to A$ ("ex-falso-quodlibet") for prime non-Δ prime formulas.
- Usual axioms of Heyting arithmetic, adapted to token types:

$$A(\mathit{tt}) \to A(\mathit{ff}) \to A(a),$$
$$A(*) \to A(0) \to \forall_{a^*,b^*}(A(a^*) \to A(b^*) \to A(\mathrm{C}a^*b^*)) \to A(a^*).$$

- $\mathrm{atom}(\mathit{tt})$.
- $\exists_x \forall_a (a \in_\rho x \leftrightarrow A)$ for A Σ-formula (ρ an object type)
- For every constant $[\![\lambda_{\vec{x}} M]\!]$, (V), (A), (C), (D), + lfp axioms.
- Defining clauses and lfp axioms for $\cup^{\#}_{\mathbf{N}}$, pcond, $\uparrow^{\#}_{\rho}$, $\exists$.
- The clauses defining the totality predicates G_ρ and the equivalence relations $x_1 \approx_\rho x_2$, together with their lfp axioms.

Theorem

TCF^+ *proves the density theorem and the definability theorem.*

Axioms of TCF^+

- Based on minimal logic. Define $\mathbf{F} := \mathrm{atom}(\mathrm{ff})$ ("falsum").
- $\mathbf{F} \to A$ ("ex-falso-quodlibet") for prime non-Δ prime formulas.
- Usual axioms of Heyting arithmetic, adapted to token types:

$$A(\mathrm{tt}) \to A(\mathrm{ff}) \to A(a),$$
$$A(*) \to A(0) \to \forall_{a^*,b^*}(A(a^*) \to A(b^*) \to A(\mathrm{C}a^*b^*)) \to A(a^*).$$

- $\mathrm{atom}(\mathrm{tt})$.
- $\exists_x \forall_a (a \in_\rho x \leftrightarrow A)$ for A Σ-formula (ρ an object type)
- For every constant $[\![\lambda_{\vec{x}} M]\!]$, (V), (A), (C), (D), + lfp axioms.
- Defining clauses and lfp axioms for $\cup^{\#}_{\mathbf{N}}$, pcond, $\uparrow^{\#}_{\rho}$, $\exists$.
- The clauses defining the totality predicates G_ρ and the equivalence relations $x_1 \approx_\rho x_2$, together with their lfp axioms.

Theorem

TCF^+ *proves the density theorem and the definability theorem.*

Axioms of TCF^+

- Based on minimal logic. Define $\mathbf{F} := \mathrm{atom}(\mathrm{ff})$ ("falsum").
- $\mathbf{F} \to A$ ("ex-falso-quodlibet") for prime non-Δ prime formulas.
- Usual axioms of Heyting arithmetic, adapted to token types:

$$A(\mathrm{tt}) \to A(\mathrm{ff}) \to A(a),$$
$$A(*) \to A(0) \to \forall_{a^*,b^*}(A(a^*) \to A(b^*) \to A(\mathrm{C}a^*b^*)) \to A(a^*).$$

- $\mathrm{atom}(\mathrm{tt})$.
- $\exists_x \forall_a (a \in_\rho x \leftrightarrow A)$ for A Σ-formula (ρ an object type)
- For every constant $[\![\lambda_{\vec{x}} M]\!]$, (V), (A), (C), (D), $+$ lfp axioms.
- Defining clauses and lfp axioms for $\cup^{\#}_{\mathbf{N}}$, pcond, $\uparrow^{\#}_{\rho}$, $\exists$.
- The clauses defining the totality predicates G_ρ and the equivalence relations $x_1 \approx_\rho x_2$, together with their lfp axioms.

Theorem

TCF^+ *proves the density theorem and the definability theorem.*

Conclusion, future work

- A semantical approach to type theory.
- TCF^{+} allows to study the Scott-Ershov model of partial continuous functionals and their formal neighborhoods.
- Tested for two basic theorems: density, definability
- Further case studies are necessary (e.g., adequacy).
- Program extraction from formalized proofs.